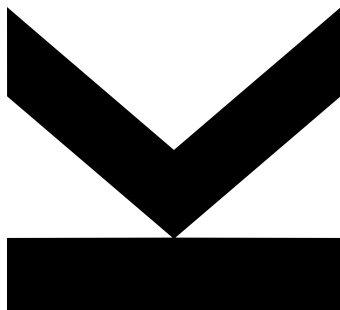


Analysis Process Modeling Notation For Business Intelligence (APMN4BI)



Doctoral Thesis

to obtain the academic degree of

Doktor der Sozial- und Wirtschaftswissenschaften

in the Doctoral Program

Sozial- und Wirtschaftswissenschaften

Author

**Dipl.-Ing. Thomas
Neuböck**

Submission

**Institute of Business
Informatics - Data &
Knowledge
Engineering**

First Supervisor

**o. Univ.-Prof.
Dipl.-Ing. Dr. Michael
Schrefl**

Second Supervisor

**Assoz.-Prof. Mag. Dr.
Christoph Schütz**

Assistant Thesis Supervisor

**Mag. Dr. Bernd
Neumayr**

November 2022

Danksagung

Der Erfolg einer Arbeit hängt immer auch von jenen Menschen ab, die einem begleiten. An dieser Stelle möchte ich mich daher bei allen Personen bedanken, die mich beim Gelingen dieser Dissertation in unterschiedlicher Form unterstützt haben.

An erster Stelle möchte ich mich bei Michael Schrefl bedanken, der mir die Möglichkeit gab, als externer Doktorand dieses langjährige Vorhaben zu starten und mich dabei mit seinem persönlichen Einsatz geduldig begleitet und betreut hat. In vielen gemeinsamen Gesprächen bekamen neue Ideen erst ihre notwendige wissenschaftliche Klarheit. Bedanken möchte ich mich an dieser Stelle insbesondere bei Christoph Schütz für die Übernahme der Zweitbetreuung meiner Arbeit.

Besonderer Dank gilt Bernd Neumayr für seine Unterstützung und Zusammenarbeit, für zahlreiche Anregungen und den wissenschaftlichen Diskurs. Bedanken möchte ich mich bei allen Kolleginnen und Kollegen der Abteilung Data & Knowledge Engineering des Institutes für Wirtschaftsinformatik für den Gedankenaustausch und für die gemeinsame Projektarbeit. Insbesondere möchte ich mich an dieser Stelle aber auch bei Margit Brandl bedanken, die mir das Leben als externer Doktorand in zahlreichen organisatorischen Angelegenheiten auf unschätzbare Weise erleichtert hat.

Zahlreiche Ideen zu dieser Dissertation stammen aus meiner jahrelangen Arbeit in Kundenprojekten. Im Besonderen wurde ich bei der Bereitstellung von Anwendungsfällen und bei der Durchführung von Fallstudien von Kunden unterstützt. In diesem Zusammenhang möchte ich mich insbesondere bei Martin Robausch und Karl-Heinz Bokesch von der Österreichischen Gesundheitskasse (Landesstellen Niederösterreich und Oberösterreich), bei

Gisela Karlsböck und Benjamin Herzog (KOTI Kobra GmbH) und bei Peter Jaskot, Sascha Juppe und Richard Kapun von der Österreichischen Pensionsversicherungsanstalt bedanken.

Ein besonderes Dankeschön gilt all meinen Arbeitskolleginnen und Arbeitskollegen der Firma solvistas. Der besondere Teamgeist, der freundschaftliche Umgang und der vorhandene Freiraum, eigene Ideen einzubringen, schafft den Nährboden für Kreativität und Innovation. Die langjährige Zusammenarbeit in Kunden- und Forschungsprojekten führte zu wichtigen Anregungen und Ideen für diese Dissertation.

Seit dem Start des Forschungsprojektes semCockpit, in dem die ersten Ideen zu meinem Vorhaben einer externen Dissertation entstanden, sind nun rund zehn Jahre vergangen. Den größten Teil meiner dazu aufgewendeten Arbeitszeit entfiel auf meine Freizeit—wertvolle Zeit, die vor allem auch von meiner Familie geopfert wurde. Der größte Dank für Verständnis und Bestärkung geht daher an meine Frau Astrid, an unsere beiden mittlerweile erwachsenen Kinder Eva und Joachim sowie an meine Eltern, im Besonderen an meine inzwischen verstorbene Mutter.

Kurzfassung

In vielen Unternehmen und Organisationen stellt Business Intelligence (BI) einen etablierten Unternehmensbereich dar, der eine unverzichtbare Basis zur Entscheidungsfindung (sowohl strategische als auch operative Entscheidungen) bereitstellt. Die Business Analystin bzw. der Business Analyst untersucht Daten, die in einem Data Warehouse gesammelt und integriert sind. Unter der Nutzung von Fachwissen durchläuft sie oder er einen Analyseprozess, um Informationen für eine effektive und effiziente Entscheidungsfindung zu erhalten. Ein Analyseprozess kann als Folge von Abfragen betrachtet werden, die schrittweise ausgeführt werden. Der Unterschied zwischen zwei Abfragen ergibt den interessanten Teil, warum ein(e) Business Analyst(in) von einer zur nächsten Abfrage geht. In diesem Prozessfluss stellt der Vergleich ein wichtiges Steuerungsmittel dar.

Konzeptuelle Modelle sind Modelle, die einem Anwendungsgebiet am nächsten kommen. In der BI können die zu analysierenden Informationen konzeptuell als dimensionale Faktenmodelle (DFM) dargestellt werden. Es gibt aber keine BI-spezifische Notation zur Modellierung der Analyseprozesse der Business Analyst(inn)en, die als auf Fachwissen basierte Navigation durch die mit einem DFM spezifizierten Daten betrachtet werden kann. In diesem Sinne steht die Aussage "Navigation ist Wissen", welches pro-aktiv modelliert werden sollte. Mit diesem Hintergrund können folgende Erfordernisse identifiziert werden: (1) Es gibt einen Bedarf an einer konzeptuellen Modellierungssprache, um Analyseprozesse in der BI spezifizieren zu können. Die Notation muss es ermöglichen, Analyseprozesse von Business Analyst(inn)en und Fachexpert(inn)en zu dokumentieren, so dass verstecktes Fachwissen sichtbar gemacht wird. (2) Die Analyseprozessmodelle müssen als Basis zur

automatisierten Abfragegenerierung und -ausführung dienen. Eine Semi-Automatisierung von Analyseprozessen ist zur schnellen Entscheidungsfindung erforderlich.

Diese Dissertation spezifiziert eine Notation zur Modellierung von Analyseprozessen für Business Intelligence: *Analysis Process Modeling Notation for Business Intelligence (APMN4BI)*. APMN4BI stellt die mittlere Schicht (*Analyseprozessebene*) von drei Modellierungsebenen dar. Diese Schicht greift auf die *Datenschicht* (die Ebene unterhalb der Analyseprozessebene) zu, die multi-dimensionale Würfel beinhaltet, welche mit DFM spezifiziert werden. Abfrageergebnisse der Analyseprozessebene werden in die *Visualisierungs- und Aktionsschicht* (die Ebene oberhalb der Analyseprozessebene) transferiert.

APMN4BI ist eine grafische Modellierungssprache, welche die Definition von *BI Analysegraphen* bereitstellt, die Analyseprozesse auf Schemaebene spezifizieren. Die Prozessausführung erfolgt auf Instanzebene. Ein BI Analysegraph ist ein gerichteter Graph, der *Analysesituationen* als Knoten und *Navigationsoperationen* als Kanten enthält. *Analysesituationen* stellen multi-dimensionale Abfragen dar, die auf einer erweiterten Form eines DFM basieren, welches auch Prädikaten- und Kennzahlenhierarchien beinhaltet. Eine *vergleichende Analysesituation* ermöglicht das Modellieren von Vergleichen (eine Hauptaktivität in Analyseprozessen). Sie verknüpft zwei Analysesituationen (Interessenskontext und Vergleichskontext) und stellt beide über eine Score-Definition in Beziehung. Ein *Navigationsoperator* repräsentiert einen Analyseschritt, der zwei Analysesituationen (Quellanalysesituation und Zielanalysesituation) verbindet. Er nimmt die Information der Quelle, führt – abhängig vom Operator selbst und seinen Parametern – Änderungen durch und transferiert die resultierende Information an das Ziel. Der semantische Unterschied zwischen beiden Analysesituationen wird sichtbar. Variablen können verwendet werden, um Benutzereingaben zur Ausführungszeit zu erzwingen. *Navigationswächter* bieten zusätzliche Steuerungsmöglichkeiten im Analyseprozess. Eine *zusammengesetzte Analysesituation* wird verwendet, um Analysesituationen zu gruppieren, die in einem Zuge instanziiert werden.

Die *Ausführung* von APMN4BI-Modellen basiert auf relationalen Datenbankschemen. DFMs werden als Sternenschemen umgesetzt, angereichert um zusätzliche Metadaten. Nach optionaler Benutzereingabe (umgesetzt über Variablen) werden Analysesituationen in SQL-Abfragen übersetzt, die ausgeführt werden können (Instanziierung von Analysesituationen). In diesem Kontext kann eine Navigationsoperation als eine Abfragetransformation betrachtet werden.

Zur Evaluierung der vorliegenden Arbeit werden verwandte Ansätze mit APMN4BI verglichen, insbesondere hinsichtlich Ausdrucksstärke. Zweitens werden reale Analyseaufgaben für Fallstudien herangezogen, um die Verwendbarkeit der Modellierungsnotation zu beurteilen und die Abfragegenerierung und -ausführung zu demonstrieren.

Abstract

In many companies and organizations, business intelligence (BI) is now well-established, providing an indispensable basis for decision making (strategic as well as operational decisions). The business analyst explores data collected and integrated in data warehouses. By applying expert knowledge, they perform an analysis process to obtain interesting information for effective and efficient decision making. An analysis process can be considered as a sequence of queries that are executed step by step. The difference between two queries represents the interesting part why a business analyst navigates from a query to the next one. In this process flow, comparison is an important means of control.

Conceptual models are models closest to an application area. In BI, the information to be analyzed can be presented conceptually by dimensional fact models (DFMs). But there is no BI specific notation for modeling the analysis processes of business analysts that can be considered as an expert-knowledge-based navigation through the data specified by a DFM. In this sense, one can say “navigation is knowledge” that should be modeled proactively. With this background, one can identify the following needs: (1) There is a need for a conceptual modeling language to specify analysis processes in BI. The notation must allow to document analysis processes of business analysts and subject matter experts such that tacit expert knowledge is made visible. (2) Analysis process models must serve as a basis to automate query generation and execution. A semi-automation of analysis processes is required for rapid decision making.

This thesis specifies an *Analysis Process Modeling Notation for Business Intelligence (APMN4BI)*. APMN4BI represents the middle tier (the *analysis*

process layer) of three model layers. This layer accesses the *data layer* (the layer below the analysis process layer) that contains multi-dimensional cubes which are specified by DFM's. Query results of the analysis process layer are transferred to the *visualization and action layer* (the layer above the analysis process layer).

APMN4BI is a graphical modeling language that provides the definition of *BI analysis graphs* which specify analysis processes at schema level. Process execution is performed at instance level. A BI analysis graph is a directed graph that comprises *analysis situations* as nodes and *navigation operations* as arcs. *Analysis situations* represent multi-dimensional queries based on an extended form of a DFM that also includes predicate and measure hierarchies. A *comparative analysis situation* allows to model comparison (a main activity in analysis processes). It joins two analysis situations (a context of interest and a context of comparison) and relates both by a score definition. A *navigation operator* represents an analysis step that links two analysis situations (source analysis situation and target analysis situation). It takes the information from the source, performs modifications depending on the operator itself and its parameters, and transfers the resulting information to the target. The semantic difference between both connected analysis situations becomes visible. Variables can be used to force user input at execution time. *Navigation Guards* provide additional control options for the analysis process. A *composite analysis situation* is used to group analysis situations that have to be instantiated at once.

The *execution* of APMN4BI models is based on relational database schemas. DFMs are realized as relational star schemas enriched by additional metadata. After optional user input (implemented by variables), analysis situations are translated into SQL queries that can be executed (instantiation of analysis situations). In this context, a navigation operation can be considered as a query transformation.

To evaluate the contributions, related approaches are compared with APMN4BI, especially with respect to expressivity. Second, real analysis tasks are used for case studies to assess the usability of the modeling notation, and to demonstrate query generation and execution.

Research Environment

The inspiration for this thesis is based on my experience in the areas of business intelligence (BI), data warehousing (DWH), and data science, which I have gained over the last twenty years. Before that, I worked in the field of software development and as a trainer for software development for about fifteen years. As an architect, consultant, business analyst, and project manager for the IT company solvistas GmbH¹ with focus on BI, DWH, and data science, I acquired deep insights about the work of subject matter experts and business analysts.² For about twenty years I have been conducting long-term BI and DWH projects for customers like Austrian and German public insurance organizations or manufacturing companies.

My research approach is based on design science in information system research as presented in [41]. The conceptual modeling language APMN4BI represents a design artifact elaborated and evaluated in a search process. Problem relevance (one guideline of the design science approach of Hevner et al. [41]) for APMN4BI was consistently perceived during my work in BI and DWH projects.

From March 2011 to February 2014, solvistas GmbH and the Department of Business Informatics – Data & Knowledge Engineering (DKE) of the Johannes Kepler University Linz carried out a joint research project with short title *semCockpit* (long title: *Semantic Cockpit: An Ontology-Driven, Interactive Business Intelligence Tool for Comparative Data Analysis*). *semCockpit* was funded by the Austrian Research Promotion Agency – Österreichische

¹Since March 2020, solvistas GmbH is a member of the enterprise group solvistas Group GmbH that was founded in March 2020.

²More personal information can be obtained from the curriculum vitae on page 459.

Forschungsförderungsgesellschaft mbH (FFG). The FFG is owned by the Republic of Austria represented by the Federal Ministry for Transport, Innovation and Technology (BMVIT) and the Federal Ministry of Science, Research and Economy (BMWFW). As associated project partners, two public health insurance organizations from Austria and Germany—Oberösterreichische Gebietskrankenkasse (OÖGKK), Linz and Deutsche Angestellten Krankenkasse (DAK), Hamburg—provided use cases and field studies for this research project. Both public health insurance organizations are long-term customers of solvistas GmbH. On the part of solvistas GmbH, I took on two roles in the semCockpit project: project manager and researcher. Special synergy effects could be achieved because I also was project manager, lead architect, and business analyst in BI and DWH projects of OÖGKK and DAK.

semCockpit is based on the dimensional fact model. Multi-dimensional ontologies provide unambiguous definitions of business terms (defined concepts) which can be used in OLAP queries. Comparison was introduced by comparative cubes comprising scores. Judgement rules are defined over facts of a comparative cube. They represent knowledge about possible explanations of a striking low or high score. Already at the beginning of the semCockpit project, requirements with respect to user guidance and the need of documenting of analysis processes were recognized. Constructs for modeling analysis processes like analysis situations, navigation steps, and BI analysis graphs were developed in parallel to the project objectives of semCockpit.

In the project scope of semCockpit, several papers were published. An introduction into the project idea was given in [94]. It also emphasizes user guidance as a general need. At the *31st International Conference on Conceptual Modeling (ER 2012), Florence, Italy, October 15-18, 2012*, beside a keynote presentation of semCockpit, two papers were presented: in [5], the usage of domain ontologies as semantic dimensions in data warehouses is demonstrated, and in [90], a first approach of BI analysis graphs for multi-dimensional navigation modeling is shown. Further ontology specific considerations with respect to OLAP can be found in [93, 95, 116]. [93] describes a Datalog-based reasoning over multi-dimensional ontologies towards an ontology-based OLAP. In [95], semantic enrichment of OLAP

cubes by the use of multi-dimensional ontologies and their representation in SQL and OWL is demonstrated. Business model ontologies for representing non-numeric measures in OLAP cubes are presented in [116]. A decision-scope approach for specializing business rules in general and analysis rules in the area of data warehouses is shown in [111]. Judgement and analysis rules for ontology-driven comparative data analysis are introduced in [120]. A comprehensive presentation of project results was given at the *Third European Summer School (eBISS 2013), Schloss Dagstuhl, Wadern, Germany, July 7-12, 2013* and can be found in [91]. This paper also presents the main ideas of BI analysis graphs that led to the definition of APMN4BI.

After finishing the semCockpit project, solvistas GmbH conducted and accompanied further long-term projects with public insurance organizations in Austria and Germany. Hence, additional input for elaborating and evaluating APMN4BI could be acquired from this application area—especially from the following organizations: *Oberösterreichische Gebietskrankenkasse (OÖGKK)*, *Niederösterreichische Gebietskrankenkasse (NÖGKK)*, *IT-Services der Sozialversicherung (ITSV)*, *Hauptverband der österreichischen Sozialversicherungsträger (HVB)*, *Pensionsversicherungsanstalt (PVA)*, *Deutsche Angestellten Krankenkasse (DAK)*.³ In these organizations, I have been involved in BI and DWH projects.

Especially, input and experience from the project/program *LEICON (LEistungsCONtrolling der österreichischen Krankenversicherungsträger)* was provided for elaborating APMN4BI. LEICON was established in 2004 and has been running with a yearly work program determined by thirteen Austrian public health insurance organisations. From the beginning in 2004, solvistas GmbH accompanies this program. I have been a member of the LEICON project team since the beginning in year 2004. LEICON focuses on the effort of health insurance organisations provided for their insurants—very

³In the last years, Austria’s public insurance companies were reorganized. Since January 2020, the public health insurance companies OÖGKK and NÖGKK are members of the *Österreichische Gesundheitskasse (ÖGK)* and the HVB was renamed to *Dachverband der österreichischen Sozialversicherungsträger (DVS)*. In this thesis, I keep the old notions because the main cooperation with these public companies was performed in the context of the original organizational structure.

often, corresponding analyses are disease specific. In 2016, PVA—another Austrian public insurance organisation responsible for the administration of Austrian pensions and rehabilitations—became a new customer of solvistas GmbH with the mission to establish a completely new DWH and BI system (called *PVDWH*). Since the beginning in 2016, I have been conducting all project phases as a lead architect and business analyst. New opportunities arose for further evaluation and elaboration of APMN4BI.

In 2014, solvistas GmbH introduced a new BI and DWH system for the Austrian brush manufacturing company KOTI Kobra GmbH. As a long-term customer for more than twenty-five years, the application field was well-known to extract further use cases for developing APMN4BI and for examine it with respect to usability. I acted as a consultant and architect to introduce the new BI and DWH system which is supervised by me until today. Many years earlier (from 1992), I developed an Enterprise Resource Planning (ERP) system for KOTI Kobra (custom made software), which I maintain to this day. Thus, the business area of KOTI Kobra was well-known for me. KOTI Kobra is a subsidiary of the European company group KOTI and provided an additionally application field (different to public health insurance organizations) for elaborating and evaluating APMN4BI. Use cases of (more operational) analysis processes in brush manufacturing were extracted and presented in two publications [92, 113]. The paper [92] was presented at the *15th IFAC/IEEE/IFIP/IFORS Symposium Information Control Problems in Manufacturing (INCOM 2015), May 11-13, 2015, Ottawa, Canada*. I demonstrated how these exemplary analysis processes are modeled proactively at schema level (by analysis graph schemas) and executed at instance level. In another publication, an approach for reference modeling for data analysis was presented [113]. There, similar use cases from brush manufacturing were used to describe analysis processes by analysis graphs.

To obtain further insights with respect to applicability, in winter semester 2014 and 2015 partial prototype implementations of a modeling and execution tool for APMN4BI were performed in two courses (*IT Projekt – Wirtschaftsinformatik*) of the Johannes Kepler University Linz where solvistas GmbH acted as an industry partner. Both implementations were based

on Java and used the metamodeling tool *MetaEdit+* [78]. The students tested the prototype implementations with small use cases from the public health insurance organizations and from the area of brush manufacturing. In another project course in winter semester 2017, another group of students implemented few parts of APMN4BI modeling (in a C#/.NET environment without an existing metamodeling framework). This implementation was based on a relational definition of APMN4BI metadata. In these courses, I acted as representative of the industry partner solvistas GmbH.

This thesis consolidates and formalizes the results obtained during the research process of about ten years. As a design artifact, APMN4BI provides a conceptual domain-specific language for modeling BI analysis processes based on OLAP operations. It was not intended to develop ready-to-use modeling and execution tools. Experience and insights collected from real industrial use cases over a long time represent a profound basis to state problem relevance and to perform design evaluation of APMN4BI. The research project semCockpit provided a well-founded setting for research rigor, for extracting research contributions, and for communication of research.

From March to September, 2018, case studies for final evaluation were performed based on real analysis processes of KOTI Kobra (March to May, 2018), the project LEICON of Austria's public health insurance organisations (March to July, 2018), and PVA (June to September, 2018). Insights of this final evaluation were incorporated in the thesis. New features detected during this evaluation were used to define possible extensions of APMN4BI. During the last two to three years the formalization and presentation of APMN4BI was revised and refined based on these case studies.

Contents

Research Environment	ix
1 Introduction	1
1.1 Background	3
1.1.1 Business Intelligence and Data Warehousing	3
1.1.2 Conceptual Modeling	6
1.1.3 OLAP and OLTP	7
1.2 Motivation	9
1.3 Real Environments for Case Studies	13
1.3.1 Austrian Public Health Insurance Organizations	14
1.3.2 Brush Manufacturing	18
1.3.3 Austria’s Public Pension Insurance Organization	21
1.4 Aims and Language Design	22
1.4.1 Aims of APMN4BI	22
1.4.2 Design Criteria	24
1.5 APMN4BI in a Nutshell	28
1.6 Research Approach	30
1.7 Contributions	32
1.8 Related Work	34
1.8.1 ROLAP as a Basis for APMN4BI	34
1.8.2 Comparison to BPMN	35
1.8.3 Inspirations from WebML	36
1.8.4 Relation and Distinction to Data and Process Mining	37
1.8.5 Guided Analytics	38

1.8.6	Previous Work towards APMN4BI	40
1.8.7	Modeling and Predicting Query Behavior	41
1.8.8	Query Recommendation	43
1.8.9	Interrelated Research and Contributions	45
1.9	Outline	47
2	Preliminaries for APMN4BI	53
2.1	Running Example	54
2.2	Model Layers for APMN4BI	57
2.2.1	The Data Layer	57
2.2.2	The Analysis Process Layer	60
2.2.3	The Presentation & Action Layer	62
2.3	Goal Hierarchy	64
2.4	Enriched Dimensional Fact Model	66
2.4.1	Dimension Schemas and Instances	70
2.4.1.1	Dimension Schemas	70
2.4.1.2	Dimension Instances	79
2.4.2	Measures and Measure Predicates	86
2.4.2.1	Base Measures	86
2.4.2.2	Base Measure Predicates	88
2.4.2.3	Aggregate Measures	90
2.4.2.4	Aggregate Measure Predicates	95
2.4.2.5	Scores	96
2.4.2.6	Score Predicates	97
2.4.3	Cube Schemas and Instances	99
2.4.3.1	Cube Schemas	99
2.4.3.2	Cube Instances	102
2.4.4	Translation into a Star Schema	104
3	Analysis Situations	111
3.1	Non-comparative Analysis Situations	112
3.1.1	Dimension Qualification	112
3.1.2	Formal Definition	114

3.1.3	Graphical Representation	117
3.1.4	Translation into SQL	123
3.1.5	Discussion	126
3.2	Comparative Analysis Situations	127
3.2.1	Preliminary Remarks	127
3.2.2	Formal Definition	129
3.2.3	Graphical Representation	133
3.2.4	Translation into SQL	139
3.2.5	Discussion	144
4	Navigation Operators	149
4.1	Navigation Step	151
4.2	Operators Not Involving Comparison	157
4.2.1	Operators Changing Granularity Level	161
4.2.2	Operators Changing Dice Node	162
4.2.3	Operators Changing Slice Conditions	168
4.2.4	Operators Changing Base Measure Conditions	170
4.2.5	Operators Changing Aggregate Measures	173
4.2.6	Operators Changing Filter Conditions	176
4.2.7	Operator Changing Cube Access	180
4.3	Operators Involving Comparison	181
4.3.1	Operators Introducing Comparison	182
4.3.2	Operators Changing Comparison	188
4.3.3	Operators Dropping Comparison	198
4.4	Use Analysis Situations as Cubes	201
4.4.1	Derived Cubes	201
4.4.2	Enrichments of Cubes	206
4.4.3	Operator Using Non-Comparative Analysis Situations as Cubes	208
4.5	Navigation Steps Containing Navigation Guards	214
5	Extension to Schema Level	219
5.1	Analysis Situation Schemas	222

5.1.1	Non-Comparative Analysis Situation Schemas	222
5.1.2	Comparative Analysis Situation Schemas	230
5.1.3	Properties of Analysis Situation Schemas	234
5.2	Navigation Step Schemas	236
5.2.1	Generic Definitions	236
5.2.2	Operators Used in Navigation Guards	241
5.2.3	Type-Compliant Navigation Steps	245
5.3	Example of Navigation Step Schemas	250
5.3.1	Examples without Navigation Guards	252
5.3.2	Examples with Navigation Guards	264
5.3.3	Navigation Patterns	266
5.4	Discussion	274
6	Business Intelligence (BI) Analysis Graphs	281
6.1	Definition of BI Analysis Graphs	283
6.1.1	Named Analysis Situations	284
6.1.2	Formal Definition of BI Analysis Graphs	286
6.2	BI Analysis Graph Schemas	289
6.2.1	Named Analysis Situation Schemas	290
6.2.2	Formal Definition of BI Analysis Graph Schemas . . .	292
6.3	Instances of Analysis Graph Schemas	302
6.4	Structuring by Subgraphs	311
6.5	Composite Analysis Situation	326
6.5.1	Preliminary Definitions	327
6.5.2	Formal Definition at Schema Level	331
6.5.3	Formal Definition at Instance Level	333
6.5.4	Graphical Representation and Examples	336
6.6	Analysis Trace and Backtracking	352
6.7	Discussion	358
7	Evaluation	363
7.1	Influence of Evaluation on the Design	365
7.2	Case Studies	376

7.2.1	Brief Descriptions of the Case Studies	377
7.2.1.1	Case Study 1: LEICON	378
7.2.1.2	Case Study 2: KOTI Kobra	380
7.2.1.3	Case Study 3: PVA	382
7.2.2	Management Summaries of the Case Studies	384
7.2.2.1	Case Study 1: LEICON	384
7.2.2.2	Case Study 2: KOTI Kobra	387
7.2.2.3	Case Study 3: PVA	389
7.2.3	Consolidated Evaluation	391
7.2.3.1	Non-comparative Analysis Situations and eDFM's	391
7.2.3.2	Comparative Analysis Situations	396
7.2.3.3	Navigation Steps and Navigation Operators	398
7.2.3.4	Comparative Navigation Operators	401
7.2.3.5	Derived Cubes	403
7.2.3.6	Extensions to Schema Level	404
7.2.3.7	Organization of BI Analysis Graphs	407
7.3	Evaluation of Claimed Design Criteria	409
7.3.1	Criteria 1: Domain Specific Conceptual Modeling Lan- guage	409
7.3.2	Criteria 2: Coherent Language Design	413
7.3.3	Criteria 3: Completeness of Model Constructs	415
7.3.4	Criteria 4: Mapping to SQL	418
7.3.5	Criteria 5: Early consistency checking	420
7.3.6	Criteria 6: Reasonable Decisions Regarding Trade-offs	421
8	Conclusion and Possible Extensions	425
	List of Figures	431
	List of Tables	438
	References	440

Curriculum Vitae

459

Chapter 1

Introduction

Contents

1.1	Background	3
1.1.1	Business Intelligence and Data Warehousing	3
1.1.2	Conceptual Modeling	6
1.1.3	OLAP and OLTP	7
1.2	Motivation	9
1.3	Real Environments for Case Studies	13
1.3.1	Austrian Public Health Insurance Organizations	14
1.3.2	Brush Manufacturing	18
1.3.3	Austria's Public Pension Insurance Organization	21
1.4	Aims and Language Design	22
1.4.1	Aims of APMN4BI	22
1.4.2	Design Criteria	24
1.5	APMN4BI in a Nutshell	28
1.6	Research Approach	30
1.7	Contributions	32
1.8	Related Work	34
1.8.1	ROLAP as a Basis for APMN4BI	34

1.8.2	Comparison to BPMN	35
1.8.3	Inspirations from WebML	36
1.8.4	Relation and Distinction to Data and Process Mining	37
1.8.5	Guided Analytics	38
1.8.6	Previous Work towards APMN4BI	40
1.8.7	Modeling and Predicting Query Behavior	41
1.8.8	Query Recommendation	43
1.8.9	Interrelated Research and Contributions	45
1.9	Outline	47

In many companies and organizations, business intelligence (BI) is now well-established, providing an indispensable basis for decision making (strategic as well as operational decisions). A business analyst explores data collected and integrated in data warehouses. By applying expert knowledge, the business analyst executes an analysis process to obtain interesting information for effective and efficient decision making. An analysis process can be considered as a sequence of queries that are executed step by step. The difference between two queries is the interesting part, the reason why a business analyst navigates from a query to the next one. In this process flow, comparison serves as an important means of control.

Conceptual models are models closest to an application area. In BI, data to be analyzed can be represented conceptually as a dimensional fact model (DFM) [34]. Yet, there is no profound BI-specific notation for modeling analysis processes of business analysts that can be considered as navigation, based on experts' knowledge, through the data specified by a DFM. In this sense, one can say "navigation is knowledge" that should be modeled proactively. Against this backdrop, the need for a conceptual modeling language to specify analysis processes in BI becomes apparent, which we discovered in multiple industrial projects. This thesis presents a conceptual modeling language, the *Analysis Process Modeling Notation for Business Intelligence (APMN4BI)*.

In this chapter, after presenting the background about business intelligence and conceptual modeling in Section 1.1, the motivation of APMN4BI is given in Section 1.2. This motivation comes from industrial projects, which also are the basis for the use cases that we employ in this thesis for illustration and evaluation purposes. On the basis of the experience from various industrial projects, we conducted modeling exercises together with prospective users and modeled real-world analysis processes using APMN4BI, which are case studies that serve for the evaluation of the presented approach. Three case studies are introduced in Section 1.3. Section 1.4 presents the aims of APMN4BI and criteria for the language design. To provide some basic notions and concepts for the subsequent presentation of contributions and related work, a brief introduction into APMN4BI is given in Section 1.5. The research approach is described in Section 1.6, the research contributions of this thesis are presented in Section 1.7, and relevant related work can be found in Section 1.8.

1.1 Background

APMN4BI is designed for modeling analysis processes in business intelligence (BI). APMN4BI is a conceptual modeling language based on multi-dimensional cubes and common operations for online analytical processing (OLAP) over such cubes. This section provides background information and short historical considerations about important fields that represent a basis for APMN4BI, namely business intelligence and data warehousing, conceptual modeling, and online analytical processing (OLAP) compared to online transaction processing (OLTP) are briefly presented in the following subsections.

1.1.1 Business Intelligence and Data Warehousing

APMN4BI represents a language for modeling BI analysis processes. Hence, we give a short historical outline of business intelligence (BI) and data warehousing (DWH). Whereas BI focuses on the processes of data analysis, DWH

represents the efficient provision and organization of data used by BI analysis processes.

Nowadays, business intelligence (BI) and data warehousing are well established in many areas of life where huge amount of heterogeneous data is available that has to be analyzed to provide well-founded bases for decision making. In the late 1980s Barry Devlin and Paul Murphy developed the architecture of a “business data warehouse” [26]. Due to the raise of requirements for reporting and data analysis they propose an architecture for business and information systems that fit better to decision support environments than operational systems would do. In the 1990s two other pioneers, Bill Inmon and Ralph Kimball, shaped the notions of DWH and BI. Bill Inmon advocates a “top-down” design that starts with an enterprise-wide view of a data warehouse where data of several operational data sources are integrated [49] (usually in third normal form data models). Only then, various subject-oriented views (data marts) are derived. Ralph Kimball propagates a “bottom-up” approach that starts with data marts that are designed by dimensional modeling techniques [63, 62]—a business process oriented enterprise view is designed via Kimball’s “enterprise data warehouse bus architecture”. Star schemas are logical data models that represent a common approach to implement data marts. Advancements on Inmon’s approach can be found in [51, 50]. A flexible way to develop enterprise data warehouses was proposed by D. Linstedt [67]: Data Vault. Data Vault specifies a modeling method for a core data warehouse (see also [48]). This method was extended to a methodology (Data Vault 2.0) which adds elements of project management and agility [68].

Although business intelligence was already mentioned by Hans Peter Luhn in 1958 [69], this notion received its meaning parallel to the development of data warehouses—a brief presentation about BI can be found, e.g., in [134]. Nowadays, BI comprises functions such as reporting, online analytical processing (OLAP; a notion coined by Edgar F. Codd in 1993 [23]), analytics, data mining, business performance management, text mining, or predictive analytics. It expresses the urge to move data analysis for decision making in the hands of business users.

The modern term “data science” was coined—also in the light of the upcoming tendency towards “big data”—to express an interdisciplinary field that combines areas like statistics, information science, mathematics, and computer science. The new occupational profile of a “data scientist” was created to breed people who understand how to extract knowledge from structured and unstructured data and who can give answers to important business questions [24]. A data scientist must have knowledge in data warehousing as well as in statistical methods, machine learning methods, and visualization of data. Sometimes the term “data science” is used as a modern notion for business intelligence also comprising other established concepts like predictive analytics or deep learning.

Data warehouses have no self purpose—they are used to satisfy business requirements. On the other hand, data warehouses represent a necessary basis for business intelligence and for integrated decision support systems [74]. Whereas OLAP concerns analysis tasks of business analysts with respect to dimensional data models [62] (i.e., how to retrieve information from data marts for analysis purposes), the extract-transform-load (ETL) process refers to how data is extracted from various operational systems and how it is integrated (transformed and loaded) into a data warehouse [61] (concerning both enterprise data warehouse and data mart).

A framework that identifies the evolution, applications, and emerging research areas of business intelligence and analytics (BI & A) is provided in [21]. The authors distinguish three evolution stages and describe them by key characteristics: BI&A 1.0 corresponds to the original database management system (DBMS) based view that focuses on structured content. Data mart design, ETL, and OLAP are in the main interest of research and application areas. Since the early 2000s, the prosperity of web technologies led to BI&A 2.0 which is characterized by web-based technologies and also taking into account unstructured content. Finally, in recent years, business intelligence and analytics are occupied by mobile and sensor-based content. Subjects such as the “internet of things”, “big data”, visualization, and human-computer interaction are trends that have been influencing BI research and applications.

APMN4BI contributes to BI as a modeling language that describes OLAP-

based navigation through dimensional data. Data is provided in data warehouses and organized with respect to dimensional data models that are implemented in star schemas. The data provision and organization is realized by ETL processes. BI analysis processes are modeled in APMN4BI such that in the case of process execution, information is queried from the data warehouse tables specified by star schemas.

1.1.2 Conceptual Modeling

As APMN4BI represents a conceptual modeling language for business analysts, this subsection gives short remarks on and examples of conceptual modeling languages in general. We also show examples and considerations of conceptual modeling and conceptual modeling languages with respect to BI and DWH.

Conceptual modeling has evolved to a widespread research area that tries to fill the gap between problem space and machine space. Conceptual models describe the problem space and the solution results from users' point of view with the goal to facilitate translation into machine space (software solution) [57, 86]. Principles of conceptual modeling of information systems can be found in [97]. Historically, the entity-relationship (ER) model—introduced by P. Chen in the 1970s [22]—is one of the well-known representatives of conceptual modeling languages. The Unified Modeling Language (UML) and the Business Process Modeling Notation (BPMN) represent newer modeling languages that are standardized by the Object Management Group (OMG) [100, 99, 98]. UML can be used for conceptual modeling as a general-purpose modeling language (for both modeling static and dynamic perspectives). BPMN is used for modeling business processes conceptually [118]. With respect to data and knowledge management, Mieu et al. claim that conceptual modeling languages must offer more expressiveness than traditional modeling languages [79]. In [25], a study is presented that assesses how conceptual modeling is used in practice. A review of some current research topics on conceptual modeling (e.g., with respect to big data, ontologies, and semantics) can be found in [123].

In business intelligence and data warehousing, conceptual modeling has evolved to a key success factor, too (see, e.g., [36, 106, 10, 28]). Most work done in this area concerns the static data view, especially multi-dimensional modeling (e.g., [34, 62, 106, 70]). Golfarelli et al. propose a dimensional fact model (DFM) as a conceptual graphic notation for multi-dimensional modeling [34]. There also exists work on conceptual modeling for ETL processes either using its own notation [133] or using BPMN [27, 1]. Some approaches more or less respect dynamic aspects on the side of business analysts [108, 128, 53]. UML 2 activity diagrams are extended by a UML profile in [119] to model the relationship between business processes and data warehouses conceptually. In [35], UML is used for modeling What-If applications conceptually in the context of business intelligence—the dynamic aspects are modeled by activity diagrams (including object flows). In [75], an approach is presented to trace the evolution of conceptual models in data warehouses based on a model driven architecture. The Common Warehouse Metamodel (CWM) is an OMG standard that allows to define the interoperability between different data warehouse systems [96].

In the case of APMN4BI, we provide constructs at schema level which can be used to model BI analysis processes that query data conceptually modeled by DFM's. A BI analysis graph schema represents a modeled analysis process that can be instantiated for process execution. The conceptual model constructs are formally introduced in this thesis accompanied by graphical representations.

1.1.3 OLAP and OLTP

APMN4BI is a domain-specific conceptual modeling language based on multi-dimensional data models and online analytical processing (OLAP). Hence, we give a short introduction of OLAP and multi-dimensional view on data warehouses and compare it with online transaction processing (OLTP) used in operational information systems.

OLAP is a query approach on data warehouses to obtain information for decision making [23]. OLAP is in contrast to OLTP used in operational

databases. Whereas OLTP is application-oriented with focus on manipulating and reading a small number of data rows within transactions, OLAP is subject-oriented with focus on reading a huge number of rows. OLTP represents a main technology of operational systems and OLAP plays an important role within decision support systems based on DWH technologies.

OLAP systems perform aggregation operations like sum or average calculation on big data sets. Drill-down or roll-up operations restrict or extend data sets such that aggregation is performed at finer or coarser granularity. Drill-across operations move to an entirely different data set to be aggregated. OLTP systems primarily select or manipulate single database rows. In OLTP, transactions can be considered as small atomic items comprising one or more query and/or manipulation statements that have to be executed completely or not at all.

Whereas OLTP systems accesses databases based on conceptual entity-relationship models, access to data warehouses via OLAP is based on multi-dimensional data models. Conceptually, one can think of data access on multi-dimensional cubes, e.g., on a cube of drug prescriptions. In the case of more than three dimensions, one can think of hypercubes. Base measures (e.g., costs, quantity) represent the values of such a cube that can be aggregated. They can be filtered and aggregated along dimensions. Each dimension denotes a “direction” that can be analyzed, e.g., with respect to insurants, with respect to doctors, with respect to drugs, and with respect to the date of drug prescriptions. Dimensions are divided into levels that are hierarchically ordered with respect to granularity, e.g., the insurant dimension has “insurant” as a level with finest granularity that represent the insurants themselves, “district” is another level of the insurant dimension with coarser granularity such that each insurant belongs to a district, and “province” is another level of the insurant dimension such that each district belongs to a province. There is a general level “top” with coarsest granularity that comprises all provinces. Analogously, dimension time can be divided into levels “date”, “month”, “quarter”, “year”, and “top of time”.

A multi-dimensional OLAP query aggregates measures with respect to dimension levels, e.g., it computes the sum of costs of drug prescriptions per

insurants' province and year. An OLAP operation performs a change of the query to another query, e.g., a drill-down operation to insurants' districts changes to a finer granularity within the insurant dimension, i.e., the sum of costs of drug prescriptions per insurants' district and year is calculated. The roll-up operation can be considered as an inverse operation of drill-down. It changes to a coarser dimension level. Other OLAP operations are slicing and dicing that restricts the query of a cube to a single slice or sub-cube. Drill-across represents a change to another cube, e.g., from drug prescriptions one can change to other cubes to analyze ambulant treatments or hospitalizations.

Multi-dimensional cubes can be prepared for OLAP in specific physical representations, multi-dimensional OLAP (MOLAP), or in relational representations, relational OLAP (ROLAP). For MOLAP, specific query languages (e.g., MDX) for multi-dimensional cubes are required, whereas for ROLAP, it is sufficient to have SQL as a standard query language for relational databases. Detailed presentations about conceptual and logical data models for multi-dimensional OLAP can be found in [34, 36, 62].

Analysis processes modeled by APMN4BI accesses data of multi-dimensional cubes which are conceptually modeled by DFMs. Queries are considered as analysis situations. OLAP provides important operations that can be modeled in APMN4BI to query data. OLAP operations in APMN4BI can be considered as navigation through the data of a multi-dimensional cube. The execution model of APMN4BI is based on ROLAP and, hence, each query can be simply translated into SQL.

1.2 Motivation

Our experience from industrial BI projects shows that there is a lack of and a need for documentation of BI analysis processes in a precise, unambiguous, and understandable way for effective and efficient reuse. Based on BI experience in several application areas, especially in the application area of Austrian and German public health and pension insurance companies and organizations, and in the area of brush manufacturing, we discovered that analysis processes should be documented in a precise and unambiguous conceptual

language for better understanding and reuse. When conducting analysis processes a lot of expert knowledge is applied. However, these analysis processes involving expert knowledge are not documented in a precise and unambiguous way such that they can be reproduced and reused without reinvention. Often such process and expert knowledge is only in the mind of subject matter experts. In the best case, there exists textual documentation that is hard to understand for novices in an application area and subsequently it is hard to reproduce such analysis processes.

Business analysts (especially in the role of consultants) need a conceptual language to document processes for data analysis. Knowledge about analysis processes has to be transferred from business analysts to other business analysts for reuse and to facilitate interpretation of results. Analysis processes are performed by business analysts and/or subject matter experts. Business analysts have knowledge in collecting requirements, in analyzing and elaborating new application areas, and in applying analysis methods. Subject matter experts provide deep knowledge in their application area, e.g., medical knowledge, economic knowledge, or knowledge about production processes. A business analyst needs time to specify and understand the aims and the steps of analysis processes in a new application area she or he has not dealt with before. A business analyst who has worked on analysis processes in the area of brush manufacturing is familiar in this area but she or he needs time to understand analysis processes of public health insurance companies, if this application area is a new one for her or him. If analysis processes are documented in a precise and unambiguous language, a business analyst can apply such analysis processes immediately even she or he is a novice concerning the application area.

The APMN4BI approach has to be distinguished from data mining approaches but both complement each other. APMN4BI supports proactive modeling. Business analysts and subject matter experts like to perform data analysis processes that are easy to handle. Online analytical processing (OLAP) represents an analysis method to navigate through data in a simple way. Measures (e.g., costs) are retrieved with respect to dimensions (e.g., doctors or insurants) and aggregated along dimension hierarchies (e.g.,

hierarchy of regions) by applying simple aggregation operations (e.g., calculating sums or averages). Such analysis processes are modeled proactively by APMN4BI. Data mining, text mining, process mining, or “higher sophisticated” statistical methods are not considered in the realm of such analysis processes. They try to extract knowledge from data which does not disagree with OLAP analysis processes modeled in APMN4BI—if anything, both can be considered as complements of each other. Results and insights of OLAP analysis processes can be used as input to data mining and the result of data mining can be used to specify useful OLAP analysis processes. Whereas data mining extracts knowledge from data, APMN4BI models use knowledge to define qualitatively better processes to analyze data.

A precise, unambiguous, and understandable modeling language for BI analysis processes allows to provide exact specifications for implementing BI applications that contain such processes. Data analysis and result visualization is performed by using BI tools (OLAP tools, reporting tools). Sometimes such tools offer at least restricted options to link queries to imitate workflows. Often higher sophisticated analysis processes must even be implemented by using programming languages. An appropriate conceptual specification of analysis processes would facilitate the development of BI applications in both cases.

A conceptual modeling language for BI analysis processes gives rise to develop modeling tools that can be used to define analysis process models that also can be executed. Individual application programming can be reduced because many analysis processes need not to be implemented by programming languages but can be modeled and executed by appropriate tools. Such modeling tools serve for specification and documentation of analysis processes. The models can be used to generate queries that can be executed by other tools or by an integrated runtime engine.

If tool support for such a modeling language is provided that also includes an execution engine, there are many use cases that apply analysis processes which can be automated. For instance, after loading data into a data warehouse, standard analysis processes modeled in APMN4BI can be automatically invoked and executed to examine key indicators and to give

hints, if there are abnormalities. Afterwards, a business analyst can check data quality—again by executing analysis processes modeled in APMN4BI. As a result she or he indeed notices bad data quality or new insights are obtained, if she or he detects that striking differences in compared measure values are due to real facts.

In combination with a hierarchy of analysis goals, the documentation of analysis processes provides a systematic collection of formally specified analyses processes that can be queried in the case of specific analysis questions (analysis tasks). For instance, if the management requests information from the BI department to make decisions for achieving specific goals, business analysts from the BI department can search for suitable analysis processes in a catalog which is organized accordingly to goal hierarchies. If a suitable analysis process is found, it can be chosen for execution to obtain the information required. Hence, a goal-oriented selection and execution of analysis processes can be provided (analysis guidance).

During the elaboration of this thesis, application of APMN4BI in the context of data privacy became an additional motivational factor. The introduction of the General Data Protection Regulation (GDPR) by the European Union (EU) in 2018 challenges many companies and public organizations. Of course, BI and DWH systems are also affected by data privacy in general and by the GDPR in particular. Among others, public health insurance and pension insurance organizations (as a special source for use cases for APMN4BI) have to deal with this (partially new) situation. For instance, access to data containing personal information has to be justified and recorded. As a means of documenting analysis processes, APMN4BI can support the satisfaction of the requirement to record critical data access containing personal information. The proactive modeling approach of APMN4BI allows to specify analysis processes before they are applied on data. Such specifications can be approved before access to data comprising personal information is performed. In this sense, APMN4BI also supports the requirement to justify such data access.

Another problem concerning data privacy will arise, if aggregated data contains a small number of human beings. Usually, aggregated data is not

critical with respect to data privacy, even if concerning human beings. Problems will arise, if the number of concerned human beings (e.g., the number of patients) and other attributes representing properties of them (e.g., sex, age, and appropriate details about residence) are contained in the aggregated data, and if the number of human beings contained in the aggregation represents (in an extreme case) only one person. In this case, if the basic population with respect to these properties is also small, one could possibly identify this specific person.¹ The proactive modeling approach of APMN4BI allows that such situations can be avoided by filtering them explicitly in the analysis process, if a small amount of persons arise in the result (often a number smaller or equal three is used in this context). This represents another motivational example of using APMN4BI with respect to data privacy.

1.3 Real Environments for Case Studies

The development of the Analysis Process Modeling Notation for Business Intelligence (APMN4BI) was motivated by experience obtained in several BI and DWH projects. Most insights originate from projects with insurance companies (especially from public health and pension insurance companies) but also from manufacturing.

To elaborate and evaluate the APMN4BI approach, case studies were performed in public health and pension insurance organizations, and in the area of brush manufacturing. This section briefly describes the organizational, technical, and subject-oriented environment of these case studies.

As presented in the first subsequent subsection, Austrian public health insurance organizations provide a basis for examples and case studies for APMN4BI. Moreover, the running example presented in this thesis was defined in the context of such public health insurance organizations. The second and third subsection present the organizational, technical, and subject-

¹For instance, consider the evaluation of a disease with respect to the exact age and with respect to fine grained territories also comprising small villages. If, for example, an aggregation results in one person who is 100 years old and who lives in a small village and nobody else in this village is 100 years old, an analyst who knows the inhabitants of this village will be able to identify this person.

oriented environment of brush manufacturing and Austria's public pension insurance organization.

Final case studies for evaluation of APMN4BI were performed in 2018. The claim was raised to search for such case studies that are based on real analysis processes. From March to May, 2018, real analysis processes of KOTI Kobra (a brush manufacturer) were specified as APMN4BI models. Real analysis processes in the context of health insurance companies were taken from a project named LEICON². This case study was performed from March to July, 2018. A final case study concerns Austria's pension insurance organization (performed from June to September, 2018). Internal final reports prepared for the involved companies and organizations can be found as appendices in an external document to this thesis [87, 88, 89]. These reports were written in German language such that employees of the involved companies and organizations were able to easily verify the content. Especially, it also would have been difficult to translate specific German terms that are used by subject matter experts. The evaluation in the form of case studies was used for a final revision and refinement of APMN4BI that leads to the final version of this thesis.

1.3.1 Austrian Public Health Insurance Organizations

Use cases from Austrian public health insurance organizations were selected to elaborate and evaluate the benefits of APMN4BI in big and complex BI and data warehouse environments. The profound experience in this organizational environment for more than fifteen years gave the inspiration to develop APMN4BI. There is not a single data warehouse but there are several BI and data warehouse systems for several public insurance organizations. Documentation of analysis processes in a precise, unambiguous, and understandable formal language would be a great gain, especially for improving effectiveness and efficiency of the reuse of analysis processes.

In Austria, health, pension, and accident insurance is mainly organized as

²The name LEICON is also used in the sense of a "data warehouse product" (see below in the subsequent subsection).

mandatory insurance in the public sector (national insurance). The administration is divided into several economically autonomous public insurance organizations as self-administrations that are coordinated by an umbrella organization.

In this organizational environment, there exists BI and data warehouse platforms of several producers (SAS, IBM, Microsoft, Pentaho) and a lot of BI and data warehouse applications (called “data warehouse products”). Data is collected in several core data warehouses. Relational OLAP (ROLAP) and multi-dimensional OLAP (MOLAP) are provided for data analysis. Additionally, specific query applications are programmed individually.

The considerations in this section focus on the data of thirteen Austrian public health insurance organizations that manage data of about eight million “active” insureds (beneficiaries). There are about thirty data warehouse products developed for more than twenty years. Some of them are installed separately per insurer and others consolidate data of all insurers. Additionally, some insurers have individual data warehouse products that only are used by themselves. Each data warehouse product was established for specific purposes, for example, control of medical services, control of insurance contributions, or fraud detection issues. In the context of this complex BI/DWH landscape, an efficient reuse of analysis processes is an important goal.

The use cases presented in this thesis refer to two data warehouse products (FOKO³, LEICON⁴) that integrate data of several operative sources which provide cleared services of all public health insurers. There are several service types provided in both data warehouses (e.g., drug prescriptions, ambulant treatments, or hospitalizations). The data of FOKO is stored and used separately per insurer with the goal that each insurer can analyze its own clearing data. On the other hand, LEICON stores clearing data of all health insurers to provide disease specific analysis about all Austrian insureds. In general clearing data is available monthly or quarterly depending on the service type and clearing system.

³German product name: FOLgeKOstenanalyse

⁴German product name: LEIstungsCONtrolling

To both data warehouses, there are inquiries from public insurance companies themselves but also from external stakeholders like ministries or universities. Often there are disease specific inquiries with specific analysis questions necessary to make decisions: "Does a health program to early detection of breast cancer increase the quality of treatment?", "Is there a lack of medical specialist care (e.g., logopedics, ergotherapy, or psychotherapy) for children and youth?", "How are the tendencies of mental illness?", "Is there an adherence to guideline-recommended drug therapy for patients with chronic heart failure?", "Is there a quality increase of patients with diabetes mellitus type 2 that participate in a disease management program?" For such disease specific inquiries, mostly more than one service type has to be analyzed.

As one use case, an analysis question was selected concerning diabetes mellitus of type 2 (DM2) also known as "maturity-onset diabetes"—a use case also utilized in related work [90, 91]. DM2 represents a chronic disease with an increasing tendency (a disease of civilization). It is important to study and permanently monitor this tendency and to develop and evaluate new measures against it. Disease management programs (DMP) are measures that define optimal treatment for patients. A disease management program for DM2 (DM2 DMP) defines optimal treatment for DM2 patients. Such a DMP must be monitored and evaluated periodically to identify potentials for improvement. The DM2 DMP is annually evaluated by LEICON. Due to analysis effort, more frequent evaluations (per half a year or quarterly) are not performed. In the subsequent paragraphs, expert knowledge is presented that has to be used to analyze this disease.

There are different groups of DM2 patients: (1) DM2 patients getting solely oral antidiabetics (OAD), (2) DM2 patients getting insulin but who are not patients with diabetes mellitus type 1 (autoimmune disorder), (3) DM2 patients getting both OAD and insulin, (4) DM2 patients who do not receive yet DM2 specific medication but who have specific characteristics in their treatments toward DM2 (DM2 risk group). DM2 patients can be registered to DM2 DMP voluntarily. These patients are treated by DM2 DMP doctors who are also registered to the program voluntarily. Both DM2 DMP patients and DM2 DMP doctors have to comply with the DM2 DMP,

and both also can deregister from the program. A DM2 DMP doctor is also allowed to treat non-DM2-DMP patients. The analysis of prevalence of all DM2 patients and of the different groups is of general interest.

For evaluation of the process quality of DM2 DMP, several indicators have to be analyzed: the number of visits to general practitioners, the number of visits to ophthalmologists, the rate of recommended DM2 specific drug prescriptions (biguanide, sulfonylureas), the rate of recommended DM2 specific drug prescriptions (biguanide, sulfonylureas) for the first curative treatment, the rate of specific laboratory tests (HbA1c checks, creatinine checks, triglycerid checks, total cholesterol checks, HDL/LDL cholesterol checks).

DM2 patients can be analyzed accordingly to an optimal treatment process specified by the DM2 DMP and accordingly to the costs with respect to the different service types (drugs, ambulant doctor treatment, hospitalization, transportation, et cetera). Comparison has to be done with previous years, between non-DM2-DMP and DM2 DMP patients, between non-DM2-DMP and DM2 DMP doctors, between non-DM2-DMP patients and DM2 DMP patients treated by DM2 DMP doctors. Detailed analysis is performed with regard to regional structures of patients and doctors, with regards to age groups and sex of patients, or with regards to medical sections of doctors. Further detailed investigations can be applied with respect to service types: drug analysis with respect to ATC⁵ classification, hospitalization analysis with respect to ICD10⁶ diagnoses, or analysis of ambulant treatments with respect to insurer specific catalogs used for clearance with doctors.

For this evaluation there exists no precise and conceptual documentation of the analysis process. Results are presented in an annual report which does not contain the steps of the analysis process. These steps are either tacit human analysis actions or they are only technically available in the form of program code (SAS code) used for analysis.

Other subjects for valuable use cases concern, for instance, health care of children and youth, mental illness, patients with chronic heart failure, or early detection of breast cancer. These and the DM2 use cases provided a

⁵Anatomical Therapeutic Chemical (ATC) Classification System for drugs

⁶International Classification of Diseases

valuable basis for elaborating and evaluating APMN4BI.

For evaluation of APMN4BI, a final case study that contains two analysis processes was performed in the context of LEICON. This case study is presented in an internal report copied in [87] (Appendix A). The context of the first analysis process concerns data quality assurance. LEICON uses data of FOKO that is transferred and transformed by ETL processes. To assess the quality of the transferred data at an early stage, activities with respect to data quality assurance have to be performed before using it for further transformations and calculations. An appropriate analysis for data quality assurance is specified and documented as an APMN4BI model.

The second analysis process of the final case study comprises measures about DM2 patients of a year that are compared with previous years to detect striking differences. This analysis process uncovers considerable changes to which public health insurers have to react. Again this analysis process was documented as an APMN4BI model. Both modelled analysis processes are used for evaluation of APMN4BI. Insights of this evaluation were used to improve and refine APMN4BI.

1.3.2 Brush Manufacturing

To evaluate the APMN4BI approach with respect to universal applicability, use cases from a different application area were selected: manufacturing of brushes. As a subsidiary of the European company group KOTI, the Austrian company KOTI Kobra produces brushes of various types and with various requirements, for instance, industrial and technical brushes, strip and sealing brushes, work tool brushes, sweeping and cleaning brushes, runway brushes, hygiene brushes, or entrance brush mats. The general definition of a brush comprises everything that consists of a body material and bristles. Accordingly to the diversity of brush types, KOTI Kobra has customers of various industries, for example, automotive, airport and winter equipment, chemical industry, electronics, or food and beverage industry. Beside mass production, customized high quality products represents an important line of business. For customization, various production parameters are relevant, for

instance, base types of brushes (strip brush, roller brush, and brush discs), characteristics of body and bristle material (temperature resistance, lifetime, mechanical load, etc.), number and ordering of drill holes for bristles, or color.

KOTI Kobra is faced with analysis tasks for solving strategic and operational issues. A previous use case can be found in [92]. For about six years KOTI Kobra uses a data warehouse and BI system (based on IBM Cognos). For analysis, billing and production data (including planned and actual measures) is provided in dimension and fact tables (as star schemas) and loaded into OLAP cubes. The core data warehouse and the cubes are updated daily. A monthly analysis of profit allows rapid decisions in strategic course corrections. Daily analysis of production material also allows to handle operative issues, for instance, to guarantee the availability of material or to find alternative material. Such operative use cases are presented in [92, 113].

The billing cube comprises revenue, costs, and profit as base measures. It can be analyzed with respect to products and customers, sales representatives, and periods. Concerning products, one has to distinguish between in-house production and buying-in (intragroup acquisition or acquisition from partners), between standard products and customized products, between different types of products (e.g., hygiene brushes, entrance brush mats, industry brushes), et cetera. Customers can be analyzed with respect to industries, regions, or customer types (consumer, manufacturer, distributor).

The production material cube comprises planned and actual quantity of the material use for production, and planned and actual costs as base measures. It can be analyzed with respect to material properties, final product, customer, and provision time. Material order information is provided by another cube. Ordered and delivered quantity and material costs are base measures that can be analyzed accordingly to material properties, supplier, and order and delivery date. A cube for working steps contains time information of the production process (planned and actual production time of employees and machines. It can be analyzed with respect to the type of working step, final product, customer, production worker, production machine, and

production time.

In a monthly profit analysis, a status must be reported to the management of the KOTI group. Relative deviations have to be analyzed in detail. Months of the current year are compared with months of previous years. It is important to recognize abnormalities in product categories, or industry, customer region, and customer types. Sales representatives have to react accordingly to such analyses. In the case of increasing costs, the cubes for production material, material orders, and working steps have to be analyzed. Interesting information about the production process is passed to employees responsible for the production process or for provisioning to encourage improvements.

This environment of a brush manufacturer was used for another final case study to evaluate APMN4BI. Again, two real analysis processes of KOTI Kobra were selected and specified by APMN4BI models. In [88] (Appendix B), a copy of an internal report about this case study can be found. Both analysis processes are performed by subject matter experts of KOTI Kobra using IBM Cognos as a BI platform.

The first analysis process concerns the procurement process of material used for production. On the one side, it is important to minimize material costs (selection of suppliers with lower prices and reduction of money tied up in stocks), on the other side, material procurement has to ensure timely provision of material (reduction of the risk of delayed delivery). In this analysis process, the consumption of certain material of the last year, end products that contain such material, customers that buy such end products, and alternative material are analyzed to determine the optimal kind and amount of material that has to be ordered.

In the second analysis process, a monthly profit analysis is performed that has to be reported to the management of the KOTI group. This analysis process has to be repeated each month such that it is useful to document it. Again, the insights of both analysis processes were recorded in the internal report printed in [88] (Appendix B) and used to improve and refine APMN4BI.

1.3.3 Austria's Public Pension Insurance Organization

In autumn 2016, Austria's Public Pension Insurance Organisation—Pensionsversicherungsanstalt (PVA)—started from scratch to implement an enterprise data warehouse (abbreviated as *PVDWH*). The PVA is responsible for administration of pensions of Austria's employees. In addition to it, the organization has the goal to preserve fitness of work. Thus it is also responsible for cures and rehabilitations.

The PVDWH is implemented on the data warehouse appliance *IBM Pure-Data System for Analytics (Netezza)*. Several internal and external data sources from various applications are integrated in a core data layer. User access is provided to a separate layer comprising data marts that are generally implemented in a relational star schema. The implementation of a standard front-end BI tool is part of a current procurement process.

In this environment, a third case study could be elaborated. A specific distinctiveness, compared to the previous case studies, was that APMN4BI could be examined in the case of DWH construction from scratch. APMN4BI can be used for abstract specification of analysis processes and supports the elaboration of requirements for building a data warehouse. Moreover, it can specify analysis processes independently from specific BI tools. In this context, this case study contributed to further evaluation and elaboration of APMN4BI.

For evaluation, two use cases were selected: (1) to support for planning new rehabilitation facilities, (2) to provide evidence of effectiveness of rehabilitation with respect to retirement. In the first use case, an interactive landscape is provided to get an overview of existing rehabilitation facilities of Austria and to simulate the establishment of new facilities. Considerations such as about diseases and disabilities, and about catchment areas are of importance in a corresponding analysis process. The second use case tries to show the effectiveness of rehabilitation. Beside human wellbeing, a successful rehabilitation also has an effect on retirement age.

For both use cases, analysis processes were specified and modeled in APMN4BI. The results are summarized in an internal report that can be

found in [89] (Appendix C). Analogously to the final case studies in the previous subsections, the insights of this case study were also used to improve and refine APMN4BI.

1.4 Aims and Language Design

This section presents the aims of APMN4BI that are derived from the motivational arguments of the previous sections. Furthermore, this section introduces design criteria for APMN4BI that support the fulfillment of these aims. Moreover, we indicate strategies how the fulfillment of these design criteria were evaluated.

1.4.1 Aims of APMN4BI

Following the motivation behind APMN4BI as outlined in Section 1.2, we derive aims for the development of a conceptual analysis process notation for business intelligence (APMN4BI). In particular, APMN4BI should be

1. a precise, unambiguous, and understandable conceptual modeling language for BI analysis processes,
2. a modeling language that makes the intention of business analysts visible when performing an analysis step,
3. a modeling language for documenting BI analysis processes,
4. a modeling language for specifying BI analysis processes,
5. a modeling language to facilitate reuse of BI analysis processes,
6. a modeling language that provides a basis for implementing BI analysis processes, and
7. a modeling language that provides a basis for implementing modeling tools for APMN4BI itself and tools to automate BI analysis process execution based on APMN4BI models.

Aim 1 expresses that APMN4BI should be a precise, unambiguous, and understandable conceptual modeling language for BI analysis processes. Of course, analysis processes for BI can be described by informal natural languages. But in many cases, such verbal descriptions are imprecise and ambiguous which is also in conflict with other items of our goal list. On one side, APMN4BI should be a formal language, on the other side, it should be a conceptual language a user (e.g., a business analyst) can easily understand with respect to her or his tasks. These tasks concern data analysis processes in business intelligence which are handled by applying OLAP operations on multi-dimensional cubes. APMN4BI should be suitable to express such BI analysis processes on multi-dimensional cubes using OLAP operations.

It is important for process documentation to make individual process steps visible. Aim 2 means that APMN4BI should make visible the steps of a business analyst. APMN4BI should document OLAP and other analysis operations. Such operations indicate the difference between two queries (analysis situations). One can also consider such operations as navigation steps from one to another analysis situation. APMN4BI should make visible these navigation steps, i.e., it should make visible the intention of a business analyst. Usually, this intention is founded on expert knowledge, i.e., “tacit expert knowledge is made visible”. In this sense, we postulate “navigation is knowledge”.

One main goal of APMN4BI is to document BI analysis processes (aim 3) for auditability and reproducibility. Together with aim 1 this documentation has to be precise, unambiguous, and understandable.

Whereas aim 3 emphasizes documentation of already performed BI analysis processes, aim 4 says that APMN4BI should be usable for specifying analysis processes. This can be considered as a proactive action. APMN4BI should be a language for proactive modeling of analysis processes. Process specifications can be provided as a basis for process execution. Although an APMN4BI model should provide a specification for performing analysis processes, one has to consider an iterative development process. During process execution new insights are collected that can be used to develop new or to adapt existing process specifications.

APMN4BI should facilitate the reuse of BI analysis processes (aim 5). Whereas aim 3 sets the focus on auditability and reproducibility of a specific and already executed analysis process, aim 5 additionally expresses that analysis processes can be generalized such that they can be re-used also for similar analysis processes. A certain degree of freedom should be provided for process control. This aim augments the utilization of APMN4BI for process specification (aim 4). From a proactively modeled process specification similar analysis processes should be derived. Furthermore, such a process specifications should provide enough restrictions for meaningful user guidance.

Aim 6 means that APMN4BI models should be usable as a specification for implementing analysis processes which is inherently related to aims 1, 4, and 5. APMN4BI models should serve as specification for business analysts that perform analysis processes by standard OLAP tools. But such models should also serve as a specification for establishing analysis processes on top of standard OLAP tools (for instance, by means of reporting tools or BI platforms that allow to implement workflows) or even as a specification for implementing analysis processes using a general-purpose programming language.

Of course, it is useful to implement a modeling tool for APMN4BI. Furthermore, the implementation of a runtime environment that executes analysis processes modeled in APMN4BI should lead to semi-automatic (by support of user interaction) or even automatic execution of analysis processes without further programming efforts. APMN4BI should provide the language basis to develop such tools (aim 7).

1.4.2 Design Criteria

The aim of APMN4BI is to provide a domain specific language that can be used to model OLAP analysis processes conceptually. APMN4BI as a domain specific language is a contribution itself. It can be used to document analysis processes at a conceptual level and facilitates communication and reuse. It was not the goal of this work to develop a modeling editor and an

Table 1.1: Claimed design criteria and evaluation strategies

No.	Claimed design criteria	Evaluation strategy
1	Domain specific conceptual modeling language	case studies
2	Coherent language design (schema/instance level, visualization)	informed arguments static analysis case studies
3	Completeness of modeling constructs	case studies
4	Mapping to SQL	static analysis informed arguments
5	Early consistency checking	static analysis informed arguments
6	Reasonable decisions regarding trade-offs	informed arguments

execution runtime for APMN4BI but a properly devised language that forms the basis for developing such tools.

Table 1.1 gives a list of design criteria which are claimed to meet the overall objectives as presented in 1.4.1. This table also shows how these claims were evaluated. Accordingly to the proposed design evaluation methods in [41], case studies, informed arguments, and static analysis are used as main evaluation methods of the design criteria in Table 1.1.

APMN4BI has to meet the purpose of a domain specific language for modeling OLAP analysis processes at a conceptual level (design criteria 1). A close one-to-one correspondence between user conceptualization and model representation has to abstract from technical complexity.

Primarily, design criteria 1 supports the fulfillment of aims 1, 2, 3, and 4 presented in 1.4.1. APMN4BI as a domain specific language has to be precise, unambiguous, and understandable to model BI analysis processes conceptually at the basis of multi-dimensional cubes and OLAP operations. The intention of the business analyst has to be made visible when performing an OLAP operation. A domain specific language is convenient for documenting and specifying BI analysis processes.

In detail, a domain specific language has to provide conceptual constructs a business analyst needs to accomplish her or his analysis tasks. A user-

oriented view of queries and comparison, and the semantic difference between two queries or comparisons has to be expressed in an easily understood way. Queries and comparisons are considered as analysis situations on a conceptual model of multi-dimensional cubes. The semantic difference between two analysis situations is considered conceptually as a navigation step from one analysis situation to the other one such that a navigation operator indicates the changes performed on the source analysis situation. Such a navigation step represents the intention of the user why she or he changes from one to the other analysis situation (“navigation is knowledge”). In common BI tools, this intention/navigation is tacit. In APMN4BI, navigation is knowledge that is made visible.

The fulfillment of design criteria 1 was evaluated in discussions and interviews of business analysts and subject matter experts in the context of real business environments presented in Section 1.3 which provided the basis for case studies. As a by-product of the evaluation result, indication on highly important or less important language constructs could be acquired that also gives rise to guidelines for future tool development.

Final case studies performed in 2018 are presented in an external document as appendices to this thesis [87, 88, 89]. As described in Section 1.3, they refer to Austrian public health insurance organizations, to brush manufacturing, and to Austria’s public pension insurance organization. The insights gained from these final case studies were used in a final development cycle to elaborate further improvements for and refinements of APMN4BI (for evaluation results, see Section 7.3.1).

For easy use and understanding a coherent language design along some common meaningful design principles are suggested and uniformly applied throughout the whole language (design criteria 2). A consistent distinction between schema and instance level has to be respected. The underlying data structures (multi-dimensional cubes) have to describe data objects at instance level and data objects at instance level must belong to a specification at schema level. Also for BI analysis processes, schema and instance level have to be distinguished. At instance level, APMN4BI documents specific analysis processes and at schema level, APMN4BI specifies a type of analysis

processes that can be used to instantiate specific ones. APMN4BI has to be designed as a visual modeling language with formally clear semantics. All important constructs a business analyst is faced (analysis situations, navigation steps) have to be depicted graphically in a uniform way.

Design criteria 2 supports the fulfillment of aims 1, 3, 4, and 5. A coherent language design fosters precision, unambiguity, and understandability. The criteria facilitates the use of APMN4BI as a modeling language for documenting and specifying BI analysis processes. The claim of aim 5 to facilitate reuse can be satisfied by models at schema level. After the presentation of the APMN4BI approach, we will argue for the fulfillment of design criteria 2 in Section 7.3.2.

Design criteria 3 claims the completeness of model constructs, i.e., the language is sufficient to model expected use cases. APMN4BI is designed to model OLAP analysis processes. A user must be able to specify queries on multi-dimensional cubes, she or he must be able to compare results of two queries, the application of an OLAP operation and a comparison step must be visible in the model. The user must be able to model control options. This design criteria supports the fulfillment of aims 1, 3, and 4. The completeness of APMN4BI is evaluated by use cases in the scope of case studies introduced in Section 1.3 (for evaluation results, see Section 7.3.3).

To show a way how the conceptual constructs of APMN4BI (analysis situations, navigation) can be technically implemented and executed, analysis situations are mapped into SQL and the linkage between analysis situations must obtain precise semantics (design criteria 4). This allows to implement useful tool support as claimed in aims 6 and 7. The fulfillment of this design criteria is finally discussed in Section 7.3.4.

To facilitate the development and use of tools for specifying APMN4BI models and for executing such models (aim 7), it is useful to provide early consistency checking (design criteria 5) to avoid or at least to mitigate modeling and execution errors. The language design of APMN4BI has to foster such checks. In Section 7.3.5, the fulfillment of this design criteria will be discussed.

In the presentation of the APMN4BI approach, we will make some rea-

sonable decisions regarding trade-offs (design criteria 6). The trade-offs are between simple APMN4BI models that are easy to handle and understand (aims 1 and 6), and, for example, preparatory work which is necessary to use APMN4BI. Such trade-offs and decisions are presented and discussed in Section 7.3.6.

1.5 APMN4BI in a Nutshell

As emphasized in the previous section, APMN4BI was designed as a conceptual and domain specific modeling language for BI analysis processes. APMN4BI is based on multi-dimensional data models and OLAP operations. It is a graphical modeling language that provides the definition of *BI analysis graph schemas* and *BI analysis graphs* which specify analysis processes at schema and instance level. Analysis processes are modeled at schema level where unbound variables and parameters allow a certain degree of freedom. Hence, a BI analysis graph schema describes a set of BI analysis processes and can be considered as a type for instantiating BI analysis processes. A BI analysis graph schema represents a directed connected multi-graph that can be used to instantiate BI analysis graphs which can be considered as directed trees. At instance level, a BI analysis graph represents process execution.

As a directed connected multi-graph, a BI analysis graph schema comprises *analysis situation schemas* as vertices and *navigation step schemas* as directed edges. Analysis situation schemas can be divided into *non-comparative analysis situation schemas* and *comparative analysis situation schemas*. Both types can contain unbound variables that have to be bound at instantiation time. A navigation step schema contains a navigation operator that corresponds in the simplest case to an OLAP operation.

A *non-comparative analysis situation schema* represents a template for multi-dimensional queries based on a DFM enriched by measure hierarchies, dimensional predicates, measure predicates, and dimensional operators. It specifies a query schema comprising a schema of cubes that have to be queried, measures that have to be computed, dimension qualifications that restrict the selection of elements of a dimension and that define the granular-

ity how measures are aggregated, and an additional condition on measures that filters the computed result set.

Comparative analysis situation schemas join two non-comparative analysis situation schemas for comparison. The joined non-comparative analysis situation schemas are called context of interest and context of comparison. As measures of comparison, scores are computed. The result set can be filtered by conditions on scores.

Analysis situation schemas can be instantiated by binding all free variables. An instantiated (non-comparative/comparative) analysis situation schema is called (*non-comparative/comparative*) *analysis situation*. Analysis situations represent specific queries on cubes. In the case of non-comparative analysis situations, multi-dimensional queries are generated, whereas in the case of comparative analysis situations, two non-comparative analysis situations are joined. Scores are retrieved as measures of comparison which can be optionally used as filter conditions for restricting result sets. Both non-comparative and comparative analysis situations can be translated in SQL statements (related to ROLAP).

On one hand, cube schemas are used in the definitions of non-comparative analysis situation schemas, on the other hand, non-comparative analysis situation schemas induces cube schemas as described in Section 4.4. Also at instance level, a non-comparative analysis situation uses a cube but also a non-comparative analysis situation induces a cube. Such *derived cubes* can be considered as SQL-views. A specific navigation operator expresses that a non-comparative analysis situation is used as a cube in another non-comparative analysis situation.

A *navigation step schema* links an analysis situation schema as source and another one as target via a navigation operation. The target analysis situation schema can be derived from the source analysis situation schema by applying the navigation operation which may contain unbound parameters. An instance of a navigation step schema is called *navigation step* and is obtained by binding all free parameters. If all navigation step schemas of a BI analysis graph schema are instantiated consistently respecting the types of the source and the target, the resulting analysis graph is an instance of

this BI analysis graph schema. A navigation step schema and a navigation step represent the semantic difference between source and target. Chapter 4 gives precise definitions of all navigation operators.

Execution of navigation steps can be additionally controlled by *navigation guards*. Navigation guards are boolean expressions that examine constituents or query result sets of a navigation step's source analysis situation. If a navigation guard is evaluated to true, the navigation step and also subsequent navigation steps are performed, otherwise navigation is aborted after the corresponding source analysis situation.

Subgraphs of BI analysis graph schemas and, respectively, subgraphs of BI analysis graphs give rise for structuring (decomposition). If analysis situation schemas and navigation step schemas of a subgraph are instantiated all at once, this subgraph can be defined as a *composite analysis situation schema* which is considered as another kind of an analysis situation schema. An instantiated composite analysis situation schema is called *composite analysis situation*.

APMN4BI is a conceptual language that allows to model BI analysis graph schemas. A BI analysis graph schema represents an APMN4BI model. If a APMN4BI model is instantiated, one obtains an executable BI analysis graph.

1.6 Research Approach

The research process follows the paradigm of design science in information system research [41]. Hevner et al. present a framework and seven guidelines that should be obeyed.

The first design science research guideline concerns *design as an artifact* that requires to create useful artifacts, e.g., constructs, models, methods, or instantiations. As a viable artifact, APMN4BI represents a language construct for modeling BI analysis processes. Motivation and aims of this design artifact are given in Section 1.2 and Section 1.4. The identified design criteria can be found in 1.4.2.

The second guideline concerns *problem relevance*, which concerns the use-

ful application of a design artifact in business environments to solve relevant problems. As evaluated in case studies and based on long-term experience in the area of business intelligence and data warehousing, the need for modeling BI analysis processes as motivated in Section 1.2 represents a relevant problem area where APMN4BI can be applied effectively. In 1.4.1, relevant objectives for utilization of APMN4BI are listed.

Design evaluation of design artifacts represents the third guideline. Hevner et al. propose observational, analytical, experimental, testing, and descriptive design evaluation methods (see [41]). Case studies and field studies are observational methods. Analytical methods comprise static analysis, architecture analysis, optimization, and dynamic analysis. Controlled experiment and simulation belong to experimental evaluation methods. Testing can be done by functional testing of “black boxes” or by structural testing of “white boxes”. Informed arguments and scenarios represent descriptive design evaluation methods. The design criteria stated in 1.4.2 assure utility, quality, and efficiency of APMN4BI. Table 1.1 associates the applied design evaluation method for each design criteria.

Design science research guideline four claims to have clear and verifiable *research contributions*. The research contributes of the APMN4BI approach are listed in Section 1.7.

Research rigor as guideline five requires to apply rigorous methods in constructing and evaluating the design artifact. As presented in this thesis all language constructs are formally defined. Rigor evaluation was performed by case studies in real business environments, by informed arguments, and by static analysis.

Guideline six reveals *design as a search process*. As presented in the preface section *Research Environment*, this thesis consolidates and formalizes the results obtained during the research process of about ten years. The industrial environment (especially the environment of Austrian and German public health and pension insurance organizations and the environment of brush manufacturing), the research project *semCockpit*, and several academic project courses provided a well-founded basis in this search process.

The last design science research guideline concerns *communication of re-*

search. Results concerning APMN4BI were presented and discussed at international scientific conferences, and were published in [90], [91], [92], and [113] (formerly published under the notion “BI analysis graphs”). In the context of case studies, the APMN4BI approach was conveyed to potential users.

1.7 Contributions

The primary goal of the presented work is to define a conceptual language for modeling BI analysis processes. Although APMN4BI models can be connected to methods for presenting query results, it was not intended to develop an approach for visualization of BI results and BI user interfaces, although analysis graphs may also serve for the development of intuitive user interfaces [43].

The overall main contribution of this thesis is the provision of a *holistic approach for modeling BI analysis processes based on OLAP operations at a conceptual level*. As a domain specific language, it reflects important activities of a business analyst. Multi-dimensional queries are modeled by non-comparative analysis situations, comparison of two non-comparative analysis situations is modeled by comparative analysis situations, the application of OLAP and more sophisticated operations is represented by navigation steps that also document the differences between two consecutive analysis situations. The composition of analysis situations and navigation steps again can be considered as another type of analysis situations. The whole approach is formally defined and leads to the notions of BI analysis graph schemas (APMN4BI models) and BI analysis graphs (reflecting execution of APMN4BI models).

From this general contribution, one can identify more specific ones as presented in this section. The contributions concern the research areas of business intelligence and conceptual modeling.

First the *navigation operators* of Chapter 4 represent a contribution itself. Starting from a conceptual view of non-comparative and comparative analysis situations, they provide more subtle semantics than simple OLAP operations and offer more options to express the intention of business an-

alysts when navigating through a multi-dimensional cube. The operators themselves are constructs at a conceptual level which are visualized by symbols that associate the operators' semantics.

To define these navigation operators, an *enrichment of the DFM* has become necessary: We introduce measure hierarchies, dimension and measure predicates, predicate hierarchies, and dimensional operators. Hence, also the semantics of the DFM was enriched as an additional contribution.

Comparative analysis situations represent another contribution of APMN4BI. Conceptually, a comparative analysis situation focuses a non-comparative analysis situation as a context of interest and relates that situation to another non-comparative analysis situation as context of comparison. Comparisons can be introduced by specific navigation operators. In this way, an important activity of business analysts can be modeled.

Both non-comparative and comparative analysis situations not only use cubes but also can be considered as (derived) cubes. These cubes can be used in turn in other analysis situations. There are navigation steps that transfer the result of a source to the target, i.e., the target uses the source as a cube. This *use-as-cube operation* represents another contribution that allows conceptual modeling of re-use of query results for further queries.

A business analyst obtains the option to consider a subgraph of an analysis situation (schema) as a *composite analysis situation (schema)*. In APMN4BI a composite analysis situation schema can be modeled, if the subgraph can be instantiated at once. In this case, it must be possible to transfer all information from source to target automatically—this can be compared with automatic links in WebML (see Section 1.8.3. The notion of composite analysis situation (schema) allows to conceptually model scenarios of dependent analysis situations which, e.g., can be used for multiscale visualization [121, 122] as, for example, realized in the BI products of *Tableau Software* [124, 85]).

The specification of *APMN4BI* consistently separates schema and instance level. BI analysis processes are modeled at schema level. At instance level, execution of a BI analysis process is documented by BI analysis graphs. Analysis situation schemas and navigation step schemas are con-

structs which facilitate type checking such that navigation at schema level induces a schema-compliant navigation at instance level.

1.8 Related Work

This section presents related work. Several approaches provided inspiration for APMN4BI. Differences and similarities to APMN4BI are shown. We start with relational OLAP (ROLAP) as a basis for APMN4BI, compare our approach with the business process modeling notation (BPMN), and present inspirations from the web modeling language WebML. Relations and distinctions to data and process mining are shown. Remarks on guided analytics are given that encourage the usage of APMN4BI. Our previous work towards APMN4BI is presented separately. Comparisons to approaches for modeling and predicting query behavior and for query recommendation are described. Finally, interrelated research and contributions are presented at the end of this section.

1.8.1 ROLAP as a Basis for APMN4BI

As already mentioned in 1.1.3, OLAP is subject-oriented and provides methods for querying data warehouses on the basis of multi-dimensional data models. In contrast, OLTP is application-oriented and accesses operational databases. Hence, APMN4BI uses OLAP operations as a basis for modeling BI analysis processes.

In the simplest case, a navigation step corresponds to an OLAP operation (e.g., drill-down or roll-up). As we will see in detail in Chapter 4, APMN4BI provides more operators with subtler distinctions, for example, a drill-down in APMN4BI lists measures per dimension entry at a finer granularity, whereas move-down navigates to a single entry at a finer granularity. Moreover, there are operators that introduce comparison. Hence, navigation operators of APMN4BI are more fine-grained and more expressive with respect to business analysts' daily work than common OLAP operators.

We follow the relational OLAP (ROLAP) approach to map conceptual

APMN4BI constructs to SQL. Moreover, semantics of APMN4BI constructs is defined by SQL. The underlying conceptual dimensional fact model [34] provides a data layer for APMN4BI that is implemented as a star schema with relational database tables [62]. Analysis situations are implemented as common SQL queries. Comparative analysis situations join two non-comparative analysis situations both representing sub-queries of an outer SQL query.

Zhang et al. [135] present a specific implementation to cope with percentages in OLAP cubes (percentage cubes). Whereas OLAP cubes are designed for efficient aggregations like summations or averaging, it is difficult to calculate fractional relationships which can be considered as a kind of measure for comparison like it is used in comparative analysis situations. The authors present percentage cubes on the top of fact tables for querying percentages on measures aggregated at multiple granularity levels. Additionally, they introduce an extension of the SQL syntax for convenient retrieval of percentages and its translation into standard SQL. In spite of that this approach copes with comparison which is based on ROLAP, this thesis follows up a simpler approach (in the sense of that no specific table structures for ROLAP-cubes and no SQL extensions are introduced) as outlined briefly in the paragraph above.

1.8.2 Comparison to BPMN

The Business Process Model and Notation (BPMN) was designed to specify and document business processes in a graphical language [98, 118]. A business process is a collection of activities which are necessary to be performed to achieve a business goal. Operational systems have to support the implementation of business processes. Thus, the implementation of business processes is primarily related with OLTP. An application that implements a business process stores and reads its data to and from operational databases.

Although BPMN could also be used for modeling analysis processes, it respects an application-oriented view that does not reflect the substance of an OLAP operation and that does not make visible the navigation from a

source analysis situation to a target analysis situation with respect of the difference of both. Thus, although analysis processes could be modeled as business processes, BPMN supports an OLTP-oriented and not an OLAP-oriented view.

APMN4BI represents a conceptual modeling language—analogously to BPMN—with the focus on modeling BI analysis processes which are based on multi-dimensional queries. As a domain specific language, APMN4BI has expressiveness that cannot be provided by BPMN.

1.8.3 Inspirations from WebML

The Web Modeling Language (WebML) is a conceptual modeling language that provides a visual notation for specifying data-intensive websites [20, 19, 18, 17].⁷ WebML provides four perspectives: the structural model, the hypertext model, the presentation model, and the personalization model. The structural model represents the data content modeled by entities and relationships. Primarily, websites access and manipulate operational data, hence, they can be considered as OLTP systems. The hypertext model specifies web-pages and their linkage. It consists of a composition model that specifies the content of a page and the navigation model that expresses how pages are linked. The composition model distinguishes six types of content units: data unit, multi-data unit, index unit, filter unit, scroller unit, and direct units (for details see [19]). The navigation model comprises non-contextual and contextual links. Non-contextual links connect independent pages, whereas contextual links connect pages such that the content of the destination page depends on the content of the source page and user's interaction. One can also think of content of the source that is transferred to the target. When a page is loaded, automatic links provide automatic navigation and information transfer to other pages without user interaction. The presentation model specifies the visual layout of a web-page and the personalization model specifies user and user groups.

⁷WebML also served as an important basis for the OMG standard *Interaction Flow Modeling Language (IFML)* that defines a graphical notation for defining user interaction and front-end behavior of software systems [101].

Although WebML is suited for data navigation, it is focused on OLTP and not on OLAP systems. Nevertheless, WebML provided some important inspirations for APMN4BI. The distinction in WebML between different model perspectives inspired the suggestion for three model layers for APMN4BI. APMN4BI's *data layer* contains multi-dimensional cubes which are specified by enriched dimensional fact models (compared to the structural model of WebML that provides entities and relationships). The *analysis process layer* represents the layer where APMN4BI models are created (corresponds to the hypertext model of WebML). The analysis process layer refers to the data layer. Queries modeled in the analysis process layer access data modeled in the data layer. Results are transferred to the third layer, the *presentation & action layer* (compared to the presentation layer in WebML). In this layer, query results are visualized or actions (e.g., sending an automatic email) are performed.

Further inspirations concern the hypertext model. Content units of the composition model have a similar status as the constituents of analysis situations in APMN4BI. Contextual links of WebML can be compared with navigation steps in APMN4BI which also transfer information from a source (with possibly added user input) to a target. If no user input is required in a navigation step, it can be compared with automatic links of WebML. Navigation steps of a composite analysis situation as will be introduced in Section 6.5 represent such “automatic links”—all analysis situations contained in a composite analysis situation are instantiated at once by automatically following the included navigation steps.

1.8.4 Relation and Distinction to Data and Process Mining

APMN4BI is a domain specific language designed for proactive modeling. BI analysis processes are elaborated and specified as APMN4BI models represented by BI analysis graph schemas. Of course, it is reasonable to perform this elaboration and specification process in an iterative and exploratory way as worked out in [91]. The execution of queries and the analysis of the query

result gives insights to elaborate or improve BI analysis graph schemas. In an iterative manner, one can say, a BI analysis graph schema induces BI analysis graphs that give rise to extend and adapt the schema which again induces BI analysis graphs, and so on. Nevertheless, the main intent is to obtain a proactively modeled schema that can be used to derive instances of BI analysis processes.

Data mining (see, for example, [38]) is a method that is used to derive insights from data without proactive modeling of subject-oriented analysis processes. One tries to extract knowledge hidden in huge amounts of data. As a difference, with APMN4BI, tacit knowledge and intentions of business analysts and subject matter experts applied in BI analysis processes is made visible by incorporating it in BI analysis graph schemas.

Related to data mining, process mining analyzes data arisen from process execution. Such process data can be retrieved, for instance, from process log files. In [131], a survey of issues and approaches of workflow processes is given, in [132], a research agenda about process mining is presented. Again, whereas APMN4BI represents a proactive modeling approach, process mining tries to extract process knowledge from process data. Nevertheless, both approaches can complement each other. Instantiations of APMN4BI models can provide input for process mining and, on the other side, insights obtained from process mining can be used to specify meaningful BI analysis graph schemas.

1.8.5 Guided Analytics

Although guided analytics is not precisely defined, it is a notion used by many producers of BI tools. It relates traditional subjects of the area of BI, like reporting or dashboard development, to subjects of BI that have become popular in the last few years, like self-service BI or (visual) analytics. A common intention of guided analytics is to provide appropriate user guidance preferably by users themselves, knowledge transfer and re-usability.

This demand originates by a “culture of bottom up decision making” such that companies “give everyone the tools to access data and analysis have the

potential to make better data-driven decisions” [7]. Guided analytics has to support business analysts in knowledge exchange and re-usability: “... knowledge workers can pursue their paths of investigation, create their own persistent templates and even share their findings with others ...” [103], “... easily and quickly transfer best-practice analytics from experts to business users in order to speed decision-making by means of automating common analysis tasks, capturing best practices ...” [127], or “... through guided analytics, organizations can capture best practices in the use of information by one user or one division and guide other users or divisions on how to use the system in the same way ...” [102]. APMN4BI was designed to model proactively analysis processes at schema level such that user guidance is provided, knowledge about analysis processes is represented, and re-usability of analysis processes is increased.

In [39], a history mechanism for information visualization was investigated. Interaction histories are considered as movements through graphs that comprise application content as nodes and content transformation as edges. Nodes represent states and edges represent actions on the states. The authors distinguish three history organizations. A stack model reflects undo and redo mechanisms, time line models reflect a linear order, and branching models represent a tree structure of applications state histories. Navigation through history states, editing histories, adding metadata and annotations to histories, searching, filtering, and exporting are considered as operations on histories. In APMN4BI, BI analysis graphs represent a branching history organization where application states are analysis situations and navigation steps are transformations from one analysis situation to another one.

A taxonomy of interactive dynamics for visual analysis is presented in [40]. Heer and Shneiderman describe twelve task types for designing analytic dialogues. These task types are grouped into three high-level categories: *data & view specification*, *view manipulation*, and *process and provenance*. *Data & view specification* comprises: *Visualize* data by choosing visual encodings, *filter* out data to focus on relevant items, *sort* items to expose patterns, and *derive* values or models from source data. *View manipulation* comprises: *Select* items to highlight, filter, or manipulate them, *navigate* to examine

high-level patterns and low-level detail, *coordinate* views for linked, multi-dimensional exploration, and *organize* multiple windows and workspaces. *Process and provenance* comprises: *Record* analysis histories for revisitation, review, and sharing, *annotate* patterns to document findings, *share* views and annotations to enable collaboration, and *guide* users through analysis tasks or stories.

The taxonomy of Heer and Shneiderman represents a helpful approach for designing APMN4BI models and for embedding such models into an execution environment with respect to the three model layers (data layer, analysis process layer, presentation & action layer) presented in Section 2.2. In APMN4BI, the task types of category *Data & View Specification* should be considered in the definition of analysis situation schemas and in their visualization in the presentation layer. The specification of navigation step schemas should regard category *View Manipulation*. The elaboration of APMN4BI models represents a process that should respect category *Process and Provenance*. Especially, concerning task type *guide*, the authors state [40]: “Analysts, however, may need to develop new strategies that are formalized to guide newcomers and provide progress indicators to experts. Visual-analysis systems can incorporate guided analytics to lead analysts through workflows for common tasks.”

1.8.6 Previous Work towards APMN4BI

Based on previous work [90, 91, 92], an *Analysis Process Modeling Notation for Business Intelligence (APMN4BI)* is specified in this thesis. APMN4BI is a graphical modeling language that provides the definition of *BI analysis graphs* which specify analysis processes at schema and instance level. In Section 1.5 a concise overview of important concepts of APMN4BI is given.

We already introduced BI analysis graphs in [90]. There, we defined a simple form of non-comparative analysis situations without filter conditions and a limited set of navigation operators. Instead of BI analysis graph schemas, we specified the notion of BI analysis graph templates (including variables) for further re-use.

In [91], we extended the approach in the context of an ontology-driven BI. In this publication, we introduced comparative and composite analysis situations, and we further distinguished schema and instance level (generic analysis situations versus individual analysis situations). As a difference to the current thesis, in [91], we used an ontology-based view of dimension and measures.

Moreover, in [91], we discussed the exploratory, iterative, and incremental characteristics of an analysis process in the sense of alternating design and use phases of analysis graphs. The current thesis considers analysis processes as “objects to be modeled”, although, we still advocate an exploratory, iterative, and incremental proceeding in the elaboration of APMN4BI models.

Further on, we presented guidance, judgment, and analysis rules to provide dynamic knowledge about guidance, static knowledge about analysis situations, and actions that could be performed depending on results of analysis situations [91, 111, 120]. In the current thesis, guidance is realized by BI analysis graph schemas, judgment and analysis rules are omitted and have to be considered as parts of the presentation & action layer that will be briefly described in Section 2.2.

This thesis resumes and extends our previous work in the sense that we specify a conceptual graphical modeling language for analysis processes as a whole and in detail, and also enriched with additional model elements (e.g., navigation guards, filter list, etc.). The focus is to provide a notation for proactive modeling of analysis processes in BI as demonstrated in a manufacturing use case in [92].

1.8.7 Modeling and Predicting Query Behavior

With APMN4BI, the dynamic view of OLAP systems can be modeled at a conceptual level. There exists related work concerning conceptual modeling of the dynamic view of OLAP systems. One approach follows its own graphical notation [108]; another one uses UML [128]. In contrast to these conceptual modeling approaches, APMN4BI introduces special first class citizens (e.g., non-comparative and comparative analysis situations, navigation

guards, comparative navigation operations, etc.) and emphasizes the visibility of semantic difference between analysis situations in the sense of "navigation is knowledge" that is modeled explicitly. Thus, APMN4BI provides more expressiveness with respect to business analysts' daily work to solve analysis tasks. Secondly, APMN4BI is intended for proactive modeling of the analysis process (similarly to BPMN)—to document such processes and the underlying knowledge. An APMN4BI model is elaborated during an exploratory data analysis process and, afterwards, it is used to guide a business analyst, if she or he re-uses this well-defined process. With respect to the analysis process execution itself, the degree of freedom is restricted to the range of variables, navigation guards, and backtracking.

Sapia [108] presents a mathematical model and a graphical notation for capturing knowledge about user interaction in OLAP systems. His requirement driven approach focuses on user query behavior (dynamic view) and provides support for the conceptual design phase. Furthermore it also fosters the physical design, the implementation, and the operation phase (e.g., runtime optimization due to query prediction). The approach is session-oriented and takes into account query sequences used to solve an analysis task. Sapia introduces query prototypes comprising measures, selection and result dimensions, as well as selection and result granularity levels. Query prototypes are similar to non-comparative analysis situations in APMN4BI, whereas the latter provides more means to represent multi-dimensional queries conceptually, e.g., distinguishing between dice nodes, slice conditions, and filter conditions. Moreover, APMN4BI treats comparisons as first class citizens by introducing comparative analysis situations. Sapia also links query prototypes by OLAP operations (drill down, roll up, rotate, and focus change), however, with the goal to compute the similarity between queries, i.e., the distance of two queries based on a weighted number of operation steps. A navigation step in APMN4BI makes the difference between two analysis situations semantically visible. Sapia's approach aims at extracting user behavior to predict future query behavior (e.g., to enable predictive caching strategies [109]). In contrast, APMN4BI is used to apply knowledge of subject matter experts for proactive modeling of analysis processes to guide users (perhaps

non-experts) for similar future analysis tasks, i.e., we define and document analysis processes similar to BPMN that is used to define and document business processes. In respect thereof, APMN4BI provides more expressiveness, e.g., the visualization of navigation operations shows semantic differences, navigation guards provide additional control, or comparative analysis situations respect comparison. Implementation aspects of a modeling tool based on Sapia's approach is presented in [37, 110].

Trujillo et al. present an object oriented approach based on UML for designing data warehouses [130, 128, 129]. At a structural level, a multi-dimensional model (dimension classes, fact classes, hierarchies, etc.) is depicted as a UML class diagram. A cube class comprises initial user requirements and represents a data analysis that can be translated into an object-query language. The user query behavior is described by state and interaction diagrams. State diagrams show the object live cycle of the cube classes with all possible OLAP operations as transitions whereas interaction diagrams demonstrate the required communication between objects of the cube classes. Compared to APMN4BI, cube classes and the used UML diagrams provide less expressiveness than analysis situations and analysis graphs. The analysis process does not become visible as it is in an APMN4BI model.

In [32], a conceptual modeling approach is introduced with respect to personalizing multi-dimensional models for OLAP. The goal is to overcome the complexity of general OLAP schemas by respecting user specific requirements, contexts, and behaviors to better satisfy specific analysis needs of decision makers.

1.8.8 Query Recommendation

Jerbi et al. present a context-aware OLAP preference model for generating personalized recommendations [52, 53, 54, 55]. Analysis contexts represent an OLAP analysis comprising a fact and a dimension context (comparable to non-comparative analysis situations in APMN4BI). It is expressed by a tree structure with respect to the internal view of the data (containing both structure and value nodes). This tree structure also describes the visualiza-

tion structure of the analysis context's query result. Analysis contexts are linked via OLAP operations to OLAP analysis graphs. Based on OLAP analysis graphs recommendation scenarios are modeled and used. Jerbi et al. distinguish between three recommendation categories (respecting user preferences): (1) interactive assistance in querying multi-dimensional data (user guidance along the query specification process—primarily based on graphical languages [16, 104]); (2) anticipatory recommendations (anticipating user navigation strategy); (3) alternative recommendation (offering helpful alternative analysis nodes). In contrast, APMN4BI focuses on a user independent documentation of analysis processes (similar to BPMN) that can be used for user guidance and automatic query generation. In [73], one can find a general survey of approaches about query recommendation techniques. Giacometti et al. [33] present a recommender system based on a log investigation of former sessions. It corresponds to a collaborative filtering approach, i.e., session logs of other users are used to anticipate navigation recommendation for a current user. A general formalization for describing analytical sessions using a multi-dimensional algebra can be found in [107]. Aufaure et al. propose a probabilistic user behavior model for query predicting in analytical sessions to proactively guide users in data exploration [6]. Bentayeb and Favre present a K-means Clustering Method for recommending OLAP queries [12]. They defined a new roll-up operator based on K-means (RoK) that combines data mining and OLAP. Marcel describes several aspects to mine query logs for extracting user behavior with respect to user-centric OLAP [71]: (a) extracting profile information, (b) personalizing queries with a single user log, (c) collaborative recommendations with a multi-user log. A personalization framework for OLAP queries is described in [11]. Aligon et al. present a collaborative filtering approach for recommending OLAP sessions [3]. A deep discussion about query similarity can be found in [4]. In [2], an OLAP tool is demonstrated that assists query and session composition to support the exploratory process of data analysis.

An approach towards intensional answers to OLAP queries is presented in [72]. It is assumed that the analyst might expect some default results. To overcome the vast number of tuples of a query result (extensional answer), the

system only returns unexpected values, i.e., in intensional query answering a user obtains the interesting part of the result that is not expected.

1.8.9 Interrelated Research and Contributions

This subsection comprises several interrelated research and work that is not considered in this thesis in detail. Some contributions treat elements of APMN4BI under a different aspect or under an extended perspective. Other contributions present approaches that can be considered as to be adjacent to APMN4BI.

In Chapter 2, enriched dimensional fact models (eDFM) are introduced which represent the basis of the conceptual data view for APMN4BI and in which ideas from an ontological-driven business intelligence elaborated in the semCockpit project (see [91]) were incorporated. A reference modeling approach for data analysis is presented in [117] and [113]. Similar to eDFMs, in [117], multidimensional data models are introduced and regarded as multidimensional reference models that can be customized by adding, omitting, or redefining certain elements like, for instance, measures, dimensions, or dimension levels. Based on a reference data mart, one obtains an appropriate data mart that is tailored to the customer's specific requirements. In [113], this approach is extended to modeling and customization of analysis processes regarding elements also used in APMN4BI like analysis situations and navigation steps. Analysis graphs can be customized by adding or deselecting analysis situations and navigation steps.

As presented in Chapter 3, analysis situations represent queries based on OLAP (in particular on ROLAP). Whereas non-comparative analysis situations can be regarded as simple OLAP queries, comparative analysis situations can be considered as two non-comparative subqueries that are joined to compare both result sets by calculating scores that additionally can be filtered by score predicates. In [114] and [65], semantic OLAP patterns are introduced which are inspired by design patterns as used in object-orientated software design which were introduced by the "Gang of Four (GoF)" [31]. For semantic OLAP patterns, a description form is proposed consisting of name,

situation, solution, structure, and example. Specific comparative patterns that describe a comparison between a set of interest and a set of comparison are exemplarily presented in [114] (the set-to-base comparison pattern, the homogeneous independent-set comparison pattern, and the heterogeneous independent-set comparison pattern). Additionally, in [65], a guided query composition with semantic OLAP patterns is proposed that is based on three knowledge graphs comprising a pattern knowledge graph, a semantic schema knowledge graph, and a binding knowledge graph which are used in a pattern instantiation process. Knowledge graph OLAP is presented in [112]. It comprises multidimensional models and query operations for contextualized knowledge graphs. Traditional OLAP cube models and OLAP query operations are adapted to perform analysis over knowledge graphs.

Based on a preliminary version of the thesis, in [42, 45, 46, 44], an OLAP endpoint for data analysis using the resource description framework (RDF) is presented. Semantic web data is based on RDF which does not follow multidimensional structures and, thus, data analysis by common OLAP operations is not possible out of the box. The presented approach proposes superimposed multidimensional schemas for RDF data analysis. In [46], this approach is formalized in detail. Based on superimposed multidimensional schemas and analogously to APMN4BI, analysis situations and semantic web analysis graphs are defined. Hilal and Schuetz [43] have also demonstrated that the concept of analysis graphs may serve as the basis for intuitive user interfaces that facilitate the analysis of data sources.

APMN4BI represents a conceptual graphical language for modeling and documenting data analysis processes based on enriched dimensional fact models (eDFMs) that in turn can be considered as a conceptual graphical notation for representing of multidimensional data. As it will be demonstrated in Section 2.2, there have to be considered three model layers for APMN4BI (the data layer, the analysis process layer, and the presentation & action layer), whereby, APMN4BI itself is focused on the analysis process layer, an eDFM belongs to the data layer, and query result sets are linked to the presentation & action layer. Although the presentation of graphical elements is an important aspect, visualization does not occupy a research focus in this

thesis. Visualization for APMN4BI can be subdivided with respect to the three model layers: visualization of analysis processes (BI analysis graphs and BI analysis graph schemas), visualization of data structures (eDFM, cubes), and visualization of query results including user interaction and other actions (corresponds to the presentation & action layer). Visualization as a focus of research is treated by Morgan et al. [83, 80, 82, 84, 81]. A domain specific language for visualization (abbreviated as VizDSL) was proposed which is based on the Interaction Flow Modeling Language (IFML). VizDSL represents a model-driven approach that can be used for modeling visualization processes. In [84], a 4-layered architecture for VizDSL is presented comprising layers for data management, data navigation, presentation, and user interface. The semantics for VizDSL is formally defined in [81].

Business intelligence is used for elaborating and monitoring enterprise goals which can be subdivided into more strategic and more operative ones. Although it was not a research focus to look at this aspect in detail, Section 2.3 provides at least some remarks on goal hierarchies. In [115], a deeper treatment about formal strategy analysis with goal models can be found. This approach uses semantic web technologies to formalize strategic reports which can be integrated in more holistic analyses.

1.9 Outline

In this section, we briefly outline the subsequent chapters in this thesis that follow the introductory chapter (Chapter 1). Chapter 2 comprises preliminaries for APMN4BI. We introduce a running example that is based on real use cases from the area of Austrian public health insurance organizations. Afterwards, a brief description of a conceptual embedding for APMN4BI is given. We consider three model layers where APMN4BI represents the middle layer that is used to model analysis processes. Beside the analysis process layer, the data layer and the presentation & action layer are presented. Furthermore, as APMN4BI is regarded as a modeling language for BI analysis processes that have to solve analysis tasks which have to be performed to achieve predefined goals, the organization of goal hierarchies is discussed.

Finally, in Chapter 2, the main preliminary for APMN4BI is introduced: the enriched dimensional fact model (eDFM). It is based on a common dimensional fact model (DFM) and enriched by further conceptual constructs. Dimensions and cubes are introduced at schema level as well as at instance level. At the end, a translation into a star schema is presented. The eDFM provides a view of data at a conceptual level that represents the basis for all other APMN4BI constructs.

Analysis situations are presented in Chapter 3. They represent conceptual constructs that can be considered as queries applied on an eDFM. This chapter comprises formal definitions, graphical representations, and translation into SQL for non-comparative (Section 3.1) and comparative analysis situations (Section 3.2). Whereas non-comparative analysis situations represent common OLAP queries at a conceptual level, comparative analysis situations introduce comparison that combine two contexts: the context of interest and the context of comparison. The translation into SQL provides precise semantics for non-comparative and comparative analysis situations.

A core aspect of APMN4BI represents the definition of navigation operators which are introduced in Chapter 4. Navigation operators are used to define navigation steps including a source and a target analysis situation. A generic formal definition of navigation steps (including navigation guards) can be found in Section 4.1. Afterwards, the formal definitions and graphical representations of all navigation operators are defined in groups. Section 4.2 introduces operators that do not involve comparison, whereas in Section 4.3 operators are defined that involve comparative analysis situations. A specific non-comparative navigation operator is presented in a separate section (Section 4.4). This operator takes a non-comparative analysis situation and uses it as a derived cube (conceptually represented as an eDFM) in the target analysis situation. Although navigation guards are a part of the generic definition of navigation steps, they are demonstrated in detail in the last section of Chapter 4.

The design of APMN4BI respects a consistent separation between instance and schema level. Whereas Chapter 3 and Chapter 4 do not consider this separation, in Chapter 5, the extension of APMN4BI to schema level is

introduced. One main characteristic represents the use of free variables at schema level that have to be bound at instance level. Section 5.1 comprises definitions and graphical representations of non-comparative and comparative analysis situation schemas, and it also defines what it means to be an instance of an analysis situation schema yielding to the notion of analysis situations as presented in Chapter 3. Navigation step schemas are introduced generically in Section 5.2. Beside the definition of navigation step schemas, again, a definition of an instance of a navigation step schema is provided which refers to the notion of navigation steps as introduced in Chapter 4. Subsection 5.2.2 extends the usage of navigation guards at schema level by introducing universal operators that return or examine constituents of source analysis situations which are necessary to define appropriate boolean expressions for navigation guards. Type-compliant (i.e., schema-compliant) navigation steps are presented in detail in the separate Subsection 5.2.3. In this context, static type checking (checking type safety at schema level) versus dynamic type checking (checking type safety at instance level) is discussed. Examples of navigation step schemas are presented in Section 5.3 and, finally, Section 5.4 concludes Chapter 5 with a discussion, especially referring to previous work.

Chapter 6 presents BI analysis graphs and BI analysis graph schemas. A BI analysis graph represents a concrete analysis process that can be executed and that is formally defined in Section 6.1 as a directed tree comprising named analysis situations as vertices. In Section 6.2, BI analysis graph schemas are formally defined. They are represented as directed multi-graphs with named analysis situation schemas as vertices. BI analysis graph schemas can be regarded as APMN4BI models that are instantiated to BI analysis graphs. Instances of BI analysis graph schemas are introduced in Section 6.3. They are considered as BI analysis graphs that induces a graph homomorphism from it to its BI analysis graph schema. To preserve an overview in comprehensive analysis processes, BI analysis graph schemas and BI analysis graphs can be hierarchically decomposed by the concept of subgraphs. This decomposition by subgraphs is presented in Section 6.4 including graphical representations. Composite analysis situation schemas and composite ana-

lysis situations can be regarded as specific subgraphs of BI analysis graph schemas and BI analysis graphs, respectively. The instantiation and execution of composite analysis situation as a whole and in one go represents the specific feature of such a subgraph. Composite analysis situation schemas and composite analysis situations are formally defined and graphically presented in Section 6.5. A BI analysis graph induces analysis situations representing database queries which can be considered as an analysis trace with a certain temporal execution order. Section 6.6 formally introduces analysis traces and backtracking. Whereas, a BI analysis graph represents a parallel analysis trace without a specific order between the tree branches of the BI analysis graph, a serial analysis trace represents one sequence of all analysis situations of a BI analysis graph such that all tree branches are also allocated to a specific temporal order. Finally, Chapter 6 is concluded with a discussion section including further remarks on previous work.

Besides static analysis and informed arguments, use cases and case studies taken from real business environments represent an important method for evaluation in this thesis. Chapter 7 describes evaluation in detail. The final version of APMN4BI was the result of an iterative research process such that evaluation results had an impact on the subsequent design of APMN4BI. Thus in the first section of Chapter 7, we describe the whole research and evaluation process of APMN4BI and present the influence of evaluation on the design of APMN4BI. Real environments for case studies are already presented in Section 1.3 of the introductory chapter. Section 7.2 provides a brief description of final case studies of which a copy of internal reports is presented as appendices in an external document to this thesis: *Case Study 1: LEICON* [87] (Appendix A), *Case Study 2: KOTI Kobra* [88] (Appendix B), *Case Study 3: PVA* [89] (Appendix C). These final case studies were performed in cooperation with Austrian public health insurance organizations, the Austrian brush manufacturer KOTI Kobra, and the Austrian public pension organization PVA. Because of the specific nomenclature and national particularities but also to provide better communication with the involved employees of these companies, the internal reports are written in German. An English translation of the management summary of each internal report is

given in Section 7.2 in separate subsections. A consolidated view of the evaluation across all use cases and case studies is provided in Subsection 7.2.3. This consolidated evaluation refers to main constructs and main concepts of APMN4BI provided in separate subsections: non-comparative analysis situations and eDFM's, comparative analysis situations, navigation steps and navigation operators, comparative navigation operators, derived cubes, extensions to schema level, and organization of BI analysis graphs. In Section 1.4 of the introductory chapter, aims of APMN4BI are specified and design criteria that provide the fulfillment of these aims are given. These design criteria are evaluated in Section 7.3 of Chapter 7 accordingly to the predefined evaluation methods specified in Table 1.1 presented in Subsection 1.4.2 included in the introductory chapter.

The final and concise Chapter 8 provides a summary and conclusion of APMN4BI but also remarks towards possible extensions of APMN4BI. Features and ideas that were not incorporated due to reduce complexity and to keep the presentation of APMN4BI as simple as possible are presented as further advancements in this final chapter.

Chapter 2

Preliminaries for APMN4BI

Contents

2.1	Running Example	54
2.2	Model Layers for APMN4BI	57
2.2.1	The Data Layer	57
2.2.2	The Analysis Process Layer	60
2.2.3	The Presentation & Action Layer	62
2.3	Goal Hierarchy	64
2.4	Enriched Dimensional Fact Model	66
2.4.1	Dimension Schemas and Instances	70
2.4.1.1	Dimension Schemas	70
2.4.1.2	Dimension Instances	79
2.4.2	Measures and Measure Predicates	86
2.4.2.1	Base Measures	86
2.4.2.2	Base Measure Predicates	88
2.4.2.3	Aggregate Measures	90
2.4.2.4	Aggregate Measure Predicates	95
2.4.2.5	Scores	96
2.4.2.6	Score Predicates	97

2.4.3	Cube Schemas and Instances	99
2.4.3.1	Cube Schemas	99
2.4.3.2	Cube Instances	102
2.4.4	Translation into a Star Schema	104

This chapter presents preliminaries for APMN4BI. It starts with the introduction of a running example that is used throughout all subsequent chapters. This running example is a simplified use case of the case study concerning public health insurance organizations. The second section describes the embedding of APMN4BI into three model layers which are presented separately in three subsections. APMN4BI models belong to the analysis process layer that represents the focus of this thesis. The analysis process layer uses the data layer. Query results generated by analysis situations of the analysis process layer are presented in the presentation & action layer. Additionally, this layer can comprise actions that are triggered by the analysis process layer, e.g., notification of users per email about query results. The third section describes how APMN4BI can be considered with respect to goal hierarchies. Analysis tasks are induced from goals which are hierarchically organized. Because APMN4BI models solve analysis tasks such models can also be organized along goal hierarchies. In the last section, we formally present the dimensional fact model and its enrichments necessary for APMN4BI. Enriched dimensional fact models specify the data layer. The formal definitions of the last section are subsequently used for defining further constructs of APMN4BI.

2.1 Running Example

To demonstrate our approach, we introduce a simplified industrial use case originated from Austrian public health insurance organizations (a manufacturing use case can be found in [92]). The information to be analyzed concerns drug prescriptions, ambulant treatments, and hospitalizations. The data is collected in data warehouses and organized in star schemas [62] that can be used for multi-dimensional cubes. Drug prescriptions, ambulant treat-

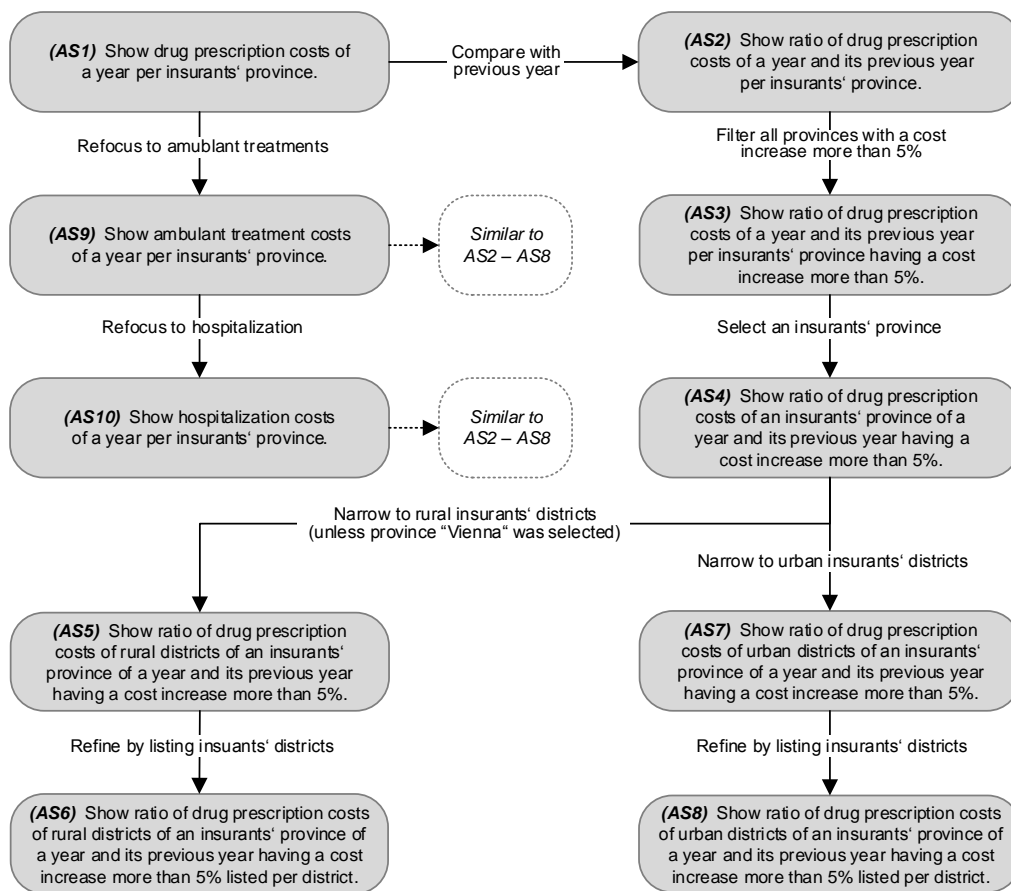


Figure 2.1: Health care use case – overview of an analysis process

ments, and hospitalizations represent facts, i.e., business events, which contain observed measures (base measures) like quantity, costs, or hospitalization days. Those facts can be evaluated with respect to several dimensions as drug, insurant, doctor, time, etc. Each dimension comprises levels that form hierarchies along which measures can be aggregated. E.g., dimension time comprises a hierarchy consisting of date, month, quarter, and year; drugs can be structured hierarchically by the international “Anatomical Therapeutic Chemical (ATC) Classification System”; insurants and doctors can be related to the regional structure of Austria, i.e., an insurant or a doctor, respectively, lives in a district and a district belongs to a province. Each dimension level can have additional properties like doctor’s age or the number of inhabitants per square kilometer of a district.

Based on these data structures, analysis processes can be defined solving analysis tasks to obtain information for decision making. In the presented application area many useful and comprehensive analysis tasks could be shown, e.g., the evaluation of disease management programs for patients with diabetes type 2 with respect to effectiveness and efficiency. To keep the running example as simple as possible, we present a part of an analysis process that evaluates exceptional cost increases.

Figure 2.1 shows an informal process flow for the analysis task of the running example. Analysis situations and navigation steps are described textually. Distinctions between schema and instance level are omitted at first. The running example presented in Figure 2.1 is used to explain various constructs of APMN4BI step by step.

The first analysis situation *AS1* of Figure 2.1 shows drug prescription costs of a selected year per province. These costs are compared by ratio with the drug prescription costs of the previous year (analysis situation *AS2*). The business analyst knows that average cost increase more than 5% can be considered as an exceptional increase. Thus, all provinces with a cost increase more than 5% are filtered in analysis situation *AS3*. The analyst selects an insurants’ province in analysis situation *AS4* and separates the cost sum into a rural and urban part (analysis situations *AS5* and *AS7*). This separation does not make sense for province “Vienna”, Austria’s cap-

ital city. Finally, for both the rural and the urban part of a province, the drug prescription costs are listed per district (analysis situations *AS6* and *AS8*). Similar analysis steps can be performed for ambulant treatment costs and hospitalization costs. These sub-processes start from analysis situations *AS9* and *AS10*, respectively. Subsequent analysis situations of them are not depicted in detail.

If one intends to describe analysis processes for evaluation of disease management programs for diabetes mellitus type 2, higher sophisticated expert knowledge must be additionally used. For example, there are different patient groups (patients who need oral anti-diabetic drugs, insulin, or both, or patients without diabetes-specific medication), there are several medical examinations and treatments that define a disease management program, there are various medical parameters that reflect the effectiveness, and there are various types of costs that quantify the efficiency of such a program. All this knowledge would have to be applied to specify an appropriate analysis process. We restrict our running example to a simple kind of cost monitoring without disease-specific considerations.

2.2 Model Layers for APMN4BI

A concise overview of the main concepts of APMN4BI was given in Section 1.5. In the current section, we present three model layers (*data layer*, *analysis process layer*, *presentation & action layer*) and the embedding of APMN4BI. The focus of this thesis lies on the analysis process layer which represents the novelty in our approach. Figure 2.2 shows an illustration of an abstract example of these model layers which are described in the subsequent subsections.

2.2.1 The Data Layer

The data layer provides a static view of data to be analyzed. It comprises multi-dimensional cubes that are modeled conceptually in the style of di-

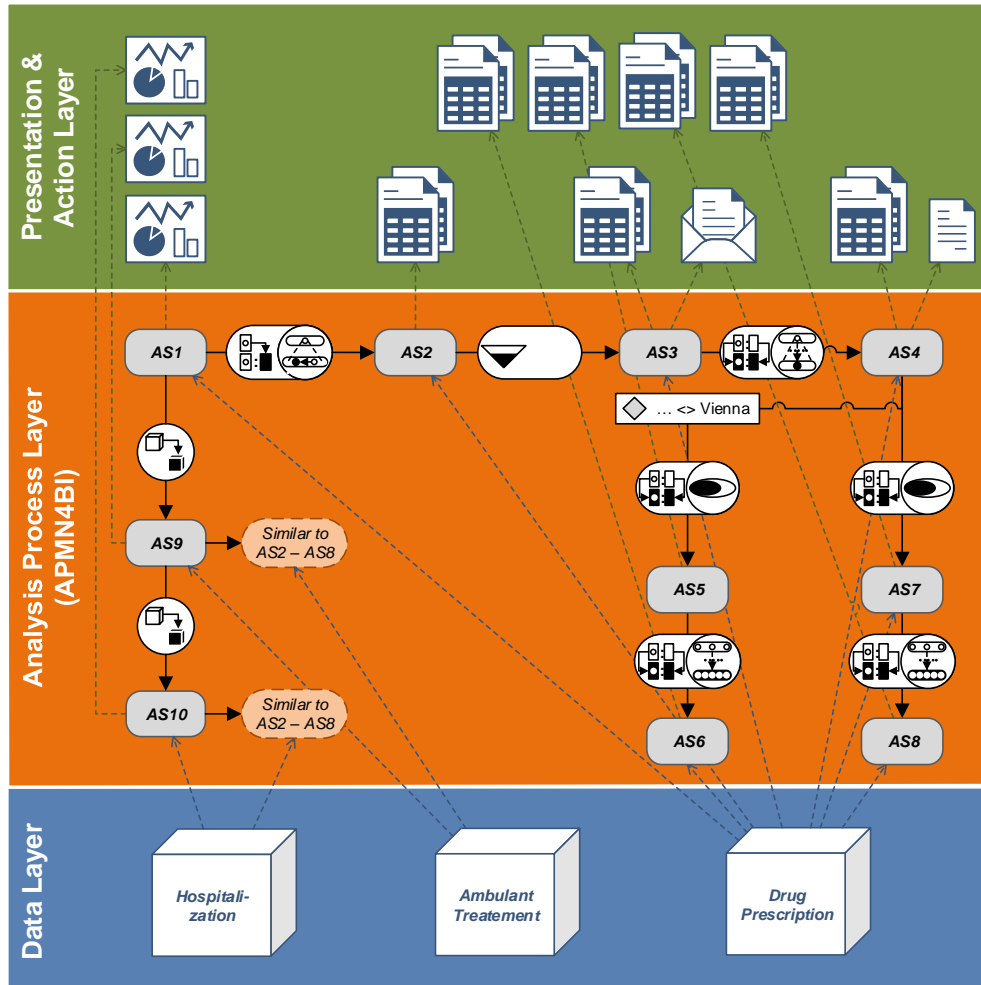


Figure 2.2: Model layers for APMN4BI

mensional fact models (DFM's) [34]. Multi-dimensional cubes¹ contain interesting information for decision-making (in most cases business events) as facts. Facts comprise base measures that can be analyzed with respect to dimensions having dimension levels with different granularity. For further reading about dimensional data modeling, we refer to [62] and, about the conceptual DFM, to [34, 36].

Dimensional fact models (as conceptual models) can be transformed into relational data models (as logical data models) and implemented in relational data bases. The underlying relational data model represents a star schema. Facts and dimensions are stored in fact tables and dimension tables. This relational implementation is a prerequisite for relational online analytical processing (ROLAP). Cubes are physically represented as fact and dimension tables which are loaded with data by extract-transform-load (ETL) processes. For further reading on ETL, we refer to [61].

Figure 2.3 on page 107 shows an enriched DFM of our health care use case. There are facts (in Figure 2.3 highlighted as red-colored rectangles) for drug prescriptions (*DrugPrescription*) and ambulant treatments (*AmbTreatment*) both comprising base measures *quantity* and *costs*. Fact *Hospitalization* contains base measures *days* and *costs*. We have dimensions (in Figure 2.3 highlighted as blue-colored ellipses) *Drug*, *MedService* (medical service), *Hospital*, *Insurant*, *Doctor*, and *Time*. For instance, in dimension *Insurant*, one can analyze measures with respect to single insurants (dimension level *insurant*), to insurants' districts (dimension level *insDistrict*), or insurants' provinces (dimension level *insProvince*).

In Figure 2.3, further elements are depicted that provide enrichments of a dimensional fact model. For example, *SumOfQuantity* and *SumOfCosts* represent aggregate measures which are again used in derived aggregate measure *AvgCostsPerUnit*. As another example of enrichment, *DocInUrbanDistrict* represents a dimensional predicate for selecting members of a dimension level. In Section 2.4, this kind of a dimensional fact model—denoted as *enriched*

¹Golfarelli et al. use the term *fact schema* [34]. In this thesis, the term *multi-dimensional cube* (or simply *cube*) is preferred. Furthermore, the terms *cube schema* and *cube instance* are introduced in this chapter.

Dimensional Fact Model (eDFM)—will be formalized and explained in detail.²

At a conceptual level, a multi-dimensional cube comprises one type of facts and all dimensions connected to these facts. We use the same name for identifying a multi-dimensional cube and its facts. In a star schema implementation, the names for facts and dimensions can also be used as names for fact and dimension tables. For example, multi-dimensional cube *DrugPrescription* comprises facts (fact table) also named as *DrugPrescription* and dimensions (dimension tables) *Drug*, *Insurant*, *Doctor*, and *Time*. In Figure 2.2, the data layer is visualized in blue color at the bottom. The multi-dimensional cubes of the eDFM of Figure 2.3 are depicted in Figure 2.2 in an abstracted notation as graphical cubes named as *DrugPrescription*, *AmbTreatment*, and *Hospitalization*.

2.2.2 The Analysis Process Layer

The analysis process layer represents the core layer that emphasizes a dynamic view of data analysis. This layer is focused in the present thesis. Analysis situations (rounded rectangles in Figure 2.2) represent queries on multi-dimensional cubes of the underlying data layer. Query results of an analysis situation are transferred to the presentation & action layer. The analysis process layer of Figure 2.2 sketches the analysis process depicted in Figure 2.1. In this concise presentation, we do not distinguish between schema and instance level first of all, i.e., we do not distinguish between BI analysis graph schema and its instantiations (BI analysis graphs). Figure 2.2 only contains a rough sketch of an APMN4BI model. For clarity, details like analysis situation specifications or operation parameters are omitted—for explanation in the subsequent paragraphs, this missing information is taken from Figure 2.1.

The process flow (navigation step) from one analysis situation to another is depicted as an arrow including a circle- or oval-shaped symbol that

²Note, in Section 2.4, we will also distinguish between schema and instance level, i.e., we will distinguish between cube schemas and cube instances, and between dimension schemas and dimension instances.

represents a navigation operation. Circles represent a navigation step from a non-comparative to another non-comparative analysis situation. A pictogram symbolizes the kind of the navigation operation. In the simplest case it corresponds to an OLAP operation. In Figure 2.2, there are navigation operations from non-comparative analysis situation *AS1* to non-comparative analysis situation *AS9* and from *AS9* to non-comparative analysis situation *AS10*. The pictogram expresses that the cube, which is queried, is exchanged. Accordingly to the textual description of Figure 2.1, one navigates from a query on cube *DrugPrescription* (*AS1*) to a query on cube *AmbTreatment* (*AS9*), and from *AS9* to a query on cube *Hospitalization* (*AS10*).

The navigation operation from non-comparative analysis situation *AS1* to comparative analysis situation *AS2* is depicted by an oval symbol and introduces comparison of drug prescription costs with the previous year. The pictogram on the left side symbolizes that the query of drug prescription costs will become the context of interest (depicted as a small rectangle containing a small circle) and from this query, the business analyst obtains a similar query as context of comparison (depicted as a small black rectangle) for selecting drug prescription costs of the previous year. The circle-shaped symbol on the right side of the oval-shaped symbol expresses how the context of comparison is derived from the context of interest—the short left-directed arrow from the small unfilled circle to the small black-filled circle symbolizes the “move from a year to the previous year”.

From comparative analysis situation *AS2* to comparative analysis situation *AS3*, there is a navigation operation that restricts the result set to rows exhibiting an increase of drug prescription costs more than 5% compared with the previous years. The pictogram within the oval-shaped operator symbol expresses the filter condition which restricts the result set.

The navigation operation from comparative analysis situation *AS3* to comparative analysis situation *AS4* changes both the context of interest (figured as a small rectangle containing a small circle) and the context of comparison (figured as a small rectangle without a small circle inside it)—a correlated change—which is depicted by the left pictogram in the oval-shaped symbol (both rectangles become black). Both in the context of interest and in

the context of comparison, a specific insurers' province is selected, i.e., one “moves down” to a specific province which is symbolized by the pictogram on the left hand of the oval-shaped symbol.

From *AS4* to *AS5* and from *AS4* to *AS7*, there are further correlated changes. The first navigation narrows to rural districts and the second to urban ones. Both restrictions are symbolized by the linked ellipses (one un-filled and the other black-filled) on the right side of the oval-shaped symbols. In the case of narrowing to rural districts, a navigation guard only allows to navigate to *AS5*, if the selected province is unequal Austrian province “Vienna”.

The navigation from *AS5* to *AS6* and from *AS7* to *AS8* performs a correlated OLAP operation “drill down”. In this case, the compared drug prescription costs and the cost increase are listed per district. The drill-down operation is expressed by the pictogram within the circle of the oval-shaped symbol meaning that one navigates from a coarser to a finer granularity level (from province to district).

Each analysis situation selects data from multi-dimensional cubes which is expressed by blue dashed arrows from the data layer to the analysis process layer. In the subsequent sections and chapters, the data layer—as an enriched dimensional fact model—and the analysis process layer—as a BI analysis graph (schema)—of the running example is refined and exactly specified after introducing necessary formalisms and definitions. Each analysis situation can be considered as an SQL query on a relational star schema (ROLAP). Such an analysis graph schema as presented in the analysis process layer of Figure 2.2 represents an APMN4BI model that can be considered as proactively modeled knowledge about analysis processes of business analysts or subject matter experts. It can be used for (semi-)automated data analysis.

2.2.3 The Presentation & Action Layer

The analysis process layer represents knowledge about data analysis processes but it does not describe the usage and presentation of analysis results. This is delegated to the presentation & action layer. In Figure 2.2, green

dashed arrows from the analysis situations of the analysis process layer and to symbolized artefacts of the presentation & action layer demonstrate the transfer of query results. This thesis focuses on the analysis process layer. Thus, the presentation & action layer is only sketched superficially in this section. No formalism of this layer is provided in subsequent chapters.

Query results of analysis situations can be visualized as data grids or diagrams (e.g., bar charts or pie charts). They can be used for preparing dashboards or static reports. Dashboards are common means how information is presented in BI systems. We do not go on further discussions about visualization principles and methods. Innovative approaches like “infographics” and “story telling” can be found, e.g., in [64, 66].

Furthermore, query results of analysis situations cannot only be used for visualization but can also be used to trigger further actions. For example, emails, business processes like procurement processes, or technical processes like further data integration and preparation processes can be triggered on the basis of a query result. This optional function of triggering and/or performing actions supports the approach of active data warehousing [126, 125] that allows to automate decision tasks.

In Figure 2.2, query results from analysis situation *AS1*, *AS9*, and *AS10*, which contain drug prescription costs, ambulant treatment costs, and hospitalization costs of a specific year per insurants’ province, are transferred to dashboards. A pictogram containing various symbols for diagrams represents such a dashboard. The green dashed arrow from an analysis situation to a pictogram of the presentation & action layer expresses this result transfer. The presentation & action layer also can provide user input to the analysis process layer (not explicitly symbolized), e.g., in an interactive dashboard the user can enter a year that is transferred to analysis situation *AS1* to determine the year of which drug prescription costs are selected. Analysis situations *AS2* – *AS8* are linked by green dashed arrows to pictograms which symbolizes data grids, i.e., query results are listed in a tabular form. Analysis situation *AS3* is the source of another green dashed arrow that leads to a mail symbol. This represents an action (sending an email) that has to be performed. The second arrow from analysis situation *AS4* to the docu-

ment symbol expresses that the query result is used in a static report. In this paragraph, these described symbols of the presentation & action layer and their interpretations must be considered exemplarily and without exact formalizations—a general and complete definition of this layer is out of scope in this thesis.

2.3 Goal Hierarchy

APMN4BI is used to specify BI analysis processes that have to solve analysis tasks for measuring the achievement of goals. In this section, some remarks and further related work to the connection of APMN4BI models to systems of goals are presented. Goals can be organized in hierarchies (goals and sub-goals) and can be divided into strategic and operative ones. Although it was not the aim of this thesis to completely investigate this connection between the APMN4BI approach and goal systems, it is important to make some considerations to show possible points of embedding APMN4BI in organizational environments and to outline how APMN4BI can be used to monitor goal systems in accordance with enterprise strategies.

Analysis tasks are connected to goal hierarchies. There are analysis tasks that have to measure and analyze achievement of goals which are positioned at a higher level within the goal hierarchy. Other analysis tasks are intended to measure and analyze the achievement of goals that contribute to the achievement of subordinate goals. Goals and their ordering in hierarchies are determined by different management levels of an organization. For instance with respect to the case studies introduced in Section 1.3, goals for health care are specified at a very high level from the ministry (see [15]). These goals are further refined by public health insurance organizations. Similarly, for the manufacturing use cases, the parent company defines goals which are refined by its subsidiaries.

APMN4BI models are defined for solving analysis tasks and can be organized along given goal hierarchies. One can think of a hierarchically ordered catalog of BI analysis graph schemas. The schema selection is driven by the goals to be analyzed. Subgraphs of a BI analysis graph schema can also be

associated to subgoals within the goal hierarchy. As an example, an overall goal of the ministry is to assure sustainably a high quality and efficient health care. One subgoal of this overall goal corresponds to the evaluation of a DM2 DMP to assure a effective and efficient treatment of DM2 patients. Similarly, to assure a sustainable growth of the whole enterprise group, all subsidiaries have to analyze their monthly revenue and profit.

A balanced scorecard (BSC) represents a tool for strategic performance management. It was introduced and elaborated by Kaplan and Norton [58, 59, 60]. Starting with a vision, a strategy is developed that gives rise to formulate objectives which are monitored by defined measures. Goals and measures are usually grouped with respect to four perspectives: (1) financial perspective, (2) customer perspective, (3) internal process perspective, (4) learning & growth perspective. Within a perspective, further goal classifications are possible and, moreover, a goal can influence another goal. Such dependencies between goals can be made visible by strategy maps. To influence the achievement of goals, initiatives and actions are implemented. The measurement to check the goals is performed periodically. Over time, the basic approach of a balanced score card has been enhanced. For instance, sustainability balanced score cards also link sustainability management to business strategy [29]. Beside economic sustainability, the approach is extended to integrate social and environmental perspectives towards social and environmental sustainability. A balanced scorecard induces a goal hierarchy. Hence, APMN4BI models can be embedded in such performance measurement systems for periodically evaluation of goal achievements.

Barone et al. present a business intelligence model (BIM) that provides goals, strategies, processes, situations, influences, and indicators as concepts (see [47, 9, 56, 8, 10]). This approach tries to bridge the business-level understanding to data-oriented representations. The authors propose strategic models for business intelligence based on goal hierarchies. Goals are refined by subgoals—strategic goals are refined by operational goals. Goals can influence other goals—operational goals influence strategic goals. Situations are internal or external factors that also influence goals. On the other side, goals can also influence situations. Performance measures are modeled by indica-

tors that can be associated to a goal or to a situation. Indicators associated to a goal are also associated to a process achieving the goal. In this sense an indicator evaluates the goal and measures the associated process. Moreover, this approach also considers a distinction between schema and instance level [56] as a general design criteria of modeling frameworks. Business intelligence models represent business schemas, i.e., goals, situations, indicators, etc. are modeled at schema level and have to be instantiated. The approach of Barone et al. can be linked to APMN4BI as a formal basis for defining goal hierarchies for organizing BI analysis graph schemas.

Key Performance Indicators (KPIs) are measures used for monitoring enterprise objectives. Maté et al. [76] state that KPIs presented in dashboards are often isolated from each other, i.e., inter-relationships between KPIs are missing. The authors propose an approach that provides an integrated view of strategic business objectives and conceptual KPIs. This approach is related to that one of Barone et al. [8]. KPIs are linked to goals which are organized in goal hierarchies.

2.4 Enriched Dimensional Fact Model

OLAP cubes are subject-oriented and provide data for analysis. As already sketched in Section 2.2.1 in the context of a data layer, cubes can be conceptually modeled by dimensional fact models (DFM) [34, 36]. In this section, the concept of an *enriched Dimensional Fact Model (eDFM)* is introduced and formalized. Explanations about an eDFM are given on the basis of our running example—the corresponding eDFM diagram is depicted in Figure 2.3.

As for common DFMs, an eDFM defines facts that in many cases represent business events like drug prescriptions, ambulant treatments, or hospitalizations. Facts comprise base measures that can be used in data analysis. In our running example, we consider quantity and costs as base measures for drug prescriptions and ambulant treatments. Hospitalization can be analyzed with respect to hospitalization days and costs.

Facts are associated with dimensions. Each dimension comprises a hi-

erarchy of dimension levels. Base measures can be aggregated along this hierarchies of dimension levels. Drug prescriptions can be analyzed with respect to doctors, insurants, drugs, and time, ambulant treatments with respect to doctors, insurants, medical services, and time, and hospitalizations with respect to insurants, hospitals, and time. Thus, in our running example, we consider dimensions for doctors, insurants, hospitals, drugs, medical services, and time.

For example, the dimension for insurants comprises a hierarchy with dimension levels for insurants (at the finest granularity), districts, and provinces. This hierarchy reflects the reality that each insurant lives in an Austrian district and each district belongs to an Austrian province³. Similarly, a dimension for doctors and a dimension for hospitals can be defined comprising a hierarchy with doctors or hospitals respectively, districts, and provinces as dimension levels. Drugs can be hierarchically organized in dimension levels with respect to the Anatomical Therapeutic Chemical (ATC) classification system⁴, and a dimension for time with respect to date, month, quarter, and year. In our example, the dimension for medical services only contains one dimension level representing medical services themselves.

At instance level, specific elements of a dimension are called dimension members or dimension nodes.⁵ Each node of a dimension belongs to exactly one dimension level. For instance, a specific doctor with name Dr. No is a member of the doctor level, Upper Austria is a member of dimension doctor at province level, or May 2018 is a specific month of the time dimension at month level.

A dimension level can contain additional properties denoted as descriptive attributes. Dimension levels for insurants and doctors have a descriptive attribute for the age of insurants and doctors. Districts (as further dimension

³Austria consists of the nine provinces Vienna, Lower Austria, Burgenland, Upper Austria, Styria, Carinthia, Salzburg, Tyrol, and Vorarlberg. Each province is divided into districts. Vienna as Austria's main capital is also considered as a province.

⁴The ATC classification system comprises five levels for classifying drugs: (1) anatomical main group (the coarsest classification level), (2) therapeutic subgroup, (3) therapeutic/pharmacological subgroup, (4) chemical/therapeutic/pharmacological subgroup, (5) chemical substance. A drug itself represents the dimension level with finest granularity.

⁵In the subsequent definitions, we prefer the term dimension node.

level for insurants, doctors, and hospitals) comprise a descriptive attribute for the number of inhabitants per square kilometer. The dimension level for drugs and medical services contain a descriptive attribute for drug prices and medical service fees.

In addition to constructs of a common DFM (facts, dimensions, levels, measures, descriptive attributes), further concepts and constructs are introduced in this thesis which are considered as an enrichment of a common DFM: classification of measures in simple base measures, derived base measures, simple aggregate measures, and derived aggregate measures; additionally, scores as special measures for comparison; predicates for base measures, aggregate measures, and scores; dimensional operators defined over dimension nodes; dimensional predicates for selecting dimension nodes.

Using simple base measures and descriptive attributes, one can define derived base measures. Quantity and costs as simple base measures can be used to define a derived base measure like costs per unit (costs divided by quantity). Simple base measures, numeric descriptive attributes, and derived base measures are summarized by the term base measure. A value of a base measure is associated with exactly one fact instance. Named conditions on base measures (denoted as base measure predicates) can be used to select facts depending on the value of base measures. For example, a base measure predicate can be defined to select only those fact instances having costs per unit more than 50 Euro.

On the basis of base measures and dimension levels, one can specify simple and derived aggregate measures. Aggregation is performed on a set of fact instances. For example, the quantity and the costs of drug prescriptions of an insurants' district can be summed up yielding simple aggregate measures (sum of quantity and the sum of costs). As another example, all insurants in a district having drug prescriptions can be counted. The number of insurants represent another simple aggregate measure counting dimension nodes of the dimension level for insurants. Having aggregate measures like sum of costs and number of insurants, one can define the average costs per insurant which represents a derived aggregate measure. Similar to base measures, also for aggregate measures, one can specify conditions to filter query results with

respect to aggregated values (aggregate measure predicate). For instance, one can define an aggregate measure predicate that restricts a query result to those records having average costs per insurant more than 1000 Euro.

Scores represent measures used to compare the result of two related queries referred as the context of interest and the context of comparison. For instance, the average costs per insurant of the actual year (as context of interest) can be compared with the average costs per insurant of the previous year (as context of comparison) by calculating the ratio of both. Similarly to aggregate measure predicates, score predicates can be defined. As an example, one can define a score predicate that is true, if there is an increase of average costs per insurant in the context of interest compared with the average costs per insurant in the context of comparison.

Further enrichments of a DFM concern dimension nodes. Dimensional operators are functions that map dimension nodes of a certain dimension level to elements of a certain domain. This domain can also represent dimension nodes of the same or of another dimension level. For example, it is useful to define a dimensional operator that maps a year to its previous year or another one that maps a month of a year to the month of the previous year. Dimensional predicates are defined to select dimension nodes. For instance, one can define a dimensional predicate that selects all rural districts and all the insurants living in rural districts. Join conditions are used to combine result sets via dimension nodes for comparison. As an example, two result sets are joined with respect to the same insurants' province. A join of records of a year with records of the previous year would be another example of a join condition.

In the subsequent subsections, we provide a formalization of an eDFM. All formal concepts are considered both at schema level and at instance level. As a prerequisite, we assume to have a universe of *dimension levels*, a universe of *descriptive attributes*, a universe of *dimensional operators*, a universe of *dimension nodes*, and a universe of *simple base measures*. All these universes are pairwise disjoint and each element X of these universes has a unique name obtained by $NameOf(X)$.⁶

⁶In this thesis, we use the name $NameOf(X)$ of an element X equivalent to the element

2.4.1 Dimension Schemas and Instances

A dimension schema defines a hierarchy with dimension levels and a dimension level represents a schema for dimension members. A dimension instance comprises all dimension members of all dimension levels of a dimension schema. We use the term dimension nodes as a synonym for dimension members that represent the items that can be analyzed in a multi-dimensional cube. Properties of dimension nodes can be specified in a dimension schema as descriptive attributes associated with a dimension level.

2.4.1.1 Dimension Schemas

In our example, we would like to evaluate insurants, doctors, drugs, medical services, and hospitals. Thus, in Figure 2.3, one can recognize dimension schemas *Insurant*, *Doctor*, *Drug*, *MedService*, and *Hospital*.⁷ *Time* is a particular dimension schema that requires specific levels and conditions as we will see later.

Before introducing a formal definition of dimension schemas, prerequisite definitions of dimensional operators, dimensional predicates, and join conditions are provided. These concepts are needed in the definition of dimension schemas.

Definition 2.1. A *dimensional operator* op is a 1-ary operator with signature $op: L \rightarrow Dom$ that can be applied to dimension nodes of level L and that returns an element of domain Dom .⁸ \square

At instance level, a dimensional operator maps a dimension node to a value of certain domain (for example, to a value of a numeric domain or to a value of a string domain). It is also allowed that a dimension node is mapped to another dimension node of the same or of another dimension level, i.e.,

itself.

⁷Generally, in this thesis, a *slant* font is used to denote names at schema level (for example, for dimension schemas or dimension levels). In graphical representations (see, for instance, Figure 2.3), different font size and thickness is used to distinguish names for different items at schema level, for example, to distinguish between names for dimension schemas and names for dimension levels.

⁸Note, it is also allowed to have $Dom = L$.

a dimensional operator can also express a dimension node by another node of the same or of another dimension level. We assume that all dimensional operators can be mapped to SQL, either by existing SQL operators or by implementing user defined functions in SQL. The name of a dimensional operator op is obtained by $NameOf(op)$.

Definition 2.2. A *dimensional predicate* is a boolean expression that can comprise dimension levels, descriptive attributes, and constants. \square

At schema level, dimensional predicates are considered as boolean expressions. Although we do not define a syntax for them, one can suppose to have an SQL based notation. Comparative operators like $=$, $<>$, $<=$, $>=$, $<$, $>$, or IN , logical operators like **AND**, **OR**, or **NOT**, parentheses (and), and arithmetic expressions are possible constituents.⁹ Thus, a boolean expression used for the definition of a dimensional predicate can be considered as a part of an SQL-where-clause. At instance level, a dimensional predicate represents a 1-ary predicate over dimension nodes. Dimensional predicates are identified by names that can be used elsewhere (for example in the definition of another dimensional predicate) to refer to it. To distinguish between the name and the expression of a dimensional predicate P , we write $NameOf(P)$ for the name and $ExprOf(P)$ for the expression.¹⁰ The name obtained by $NameOf(P)$ is used equivalently to predicate P itself.

Join conditions are used to combine result sets of two queries. They are defined over qualified dimension levels and provide a prerequisite for comparisons in the sense that two result sets are combined for comparison where one result set represents a *context of interest* (abbreviated as *CoI*) and the second one is considered as a *context of comparison* (abbreviated as *CoC*). In APMN4BI, comparisons are represented by comparative analysis situations introduced in Section 3.2 of Chapter 3. A comparative analysis situation comprises a context of interest and a context of comparison that can be considered as two subqueries which are linked by a join condition.

⁹For representation of syntactical symbols (terminal symbols), we use a **typewriter** font in this thesis.

¹⁰ $ExprOf(P)$ returns the boolean expression of P such that names of other dimensional predicates contained are also resolved by there own expression.

The following definition specifies the notion of a join condition that, later in this section, will be used as a component of the definition of a dimension schema, and that will be used in Section 3.2 in the definition of a comparative analysis situation.

Definition 2.3. A *join condition* is a boolean expression defined over qualified dimension levels qualified by **CoI** and **CoC** (syntactically separated by a dot), and possibly combined with dimensional operators. \square

Again, join conditions are defined in an SQL based syntax that can be considered as SQL join conditions used in SQL-where- or SQL-on-clauses. In the APMN4BI approach, we restrict to inner joins used to construct comparison. Qualifiers **CoI** and **CoC** represent aliases that refer to both result sets that are joined (result set for *context of interest* and result set for *context of comparison*). A join condition refers to dimension levels and it describes how dimension nodes of both result sets are combined. As for dimensional predicates, we write $NameOf(J)$ for the name and $ExprOf(J)$ for the expression of a join condition J . The name $NameOf(J)$ of join condition J is used equivalently to itself. For instance, Figure 2.3 contains two join conditions: join condition with name $SameInsProvince$ and boolean expression $CoI.insProvince = CoC.insProvince$, and join condition with name $PrevYear$ and boolean expression $prevYear(CoI.year) = CoC.year$ that includes dimensional operator $prevYear$. The first example refers dimension level $insProvince$ of dimension schema $Insurant$ and the second example refers dimension level $year$ of dimension schema $Time$. In both examples, dimension levels $insProvince$ and $year$ are referred twice, on the one hand in the context of interest expressed by label **CoI** and on the other hand in the context of comparison expressed by label **CoC**.

After these preliminary definitions, one is able to specify dimension schemas which represent main parts of cube schemas. The components of a dimension schema are listed in the following definition.

Definition 2.4. A *dimension schema* $D = (Lvl_D, RollupRel_D, DescrAttr_D, LvlOfDescrAttr_D, DimOperators_D, LvlOfDimOperator_D, DimPredicates_D, LvlOfDimPredicate_D, JoinConditions_D, LvlOfJoinCondition_D)$ comprises

1. a finite set of levels $Lvls_D$ with at least two elements,
2. a relation $RollupRel_D$ (*roll-up relation*) which defines a strict total order on $Lvls_D$,
3. a finite set of descriptive attributes $DescrAttrs_D$,
4. a function $LvlOfDescrAttr_D: DescrAttrs_D \rightarrow Lvls_D$,
5. a finite set of dimensional operators $DimOperators_D$,
6. a function $LvlOfDimOperator_D: DimOperators_D \rightarrow Lvls_D$,
7. a finite set of dimensional predicates $DimPredicates_D$,
8. a function $LvlOfDimPredicate_D: DimPredicates_D \rightarrow Lvls_D$,
9. a finite set of join conditions $JoinConditions_D$, and
10. a function $LvlOfJoinCondition_D: JoinConditions_D \rightarrow Lvls_D$.

Furthermore, the following conditions on dimensional operators and dimensional predicates must hold:

- For all $op \in LvlOfDimOperator_D$ with signature $op: L \rightarrow Dom$: $LvlOfDimOperator_D(op) = L$.
- For all $P \in DimPredicates_D$, P can only comprise (beside constants) dimension levels $L \in Lvls_D$ or descriptive attributes A of L (i.e., $LvlOfDescrAttr_D(A) = L$) such that $L = LvlOfDimPredicate_D(P)$ or $(L, LvlOfDimPredicate_D(P)) \in RollupRel_D$.

Additionally, we define *top level* top_D to be the maximum of $Lvls_D$ with respect to $RollupRel_D$ and *base level* $base_D$ to be the minimum of $Lvls_D$ with respect to $RollupRel_D$. For $(L_1, L_2) \in RollupRel_D$, L_1 is a *sublevel* of L_2 (written as $L_1 \twoheadrightarrow L_2$); and L_1 is a *direct sublevel* of L_2 (written as $L_1 \rightarrow L_2$), if there is no $L \in Lvls_D$ such that $(L_1, L) \in RollupRel_D$ and $(L, L_2) \in RollupRel_D$. We also say L_1 *rolls up* to L_2 and L_1 *directly rolls up*

to L_2 respectively.

Furthermore, we define the following auxiliary functions:

- a function $DescrAttrsAtLvl_D: Lvs_D \rightarrow \mathcal{P}(DescrAttrs_D)$ that maps each level $L \in Lvs_D$ to the set of descriptive attributes $LvlOfDescrAttr_D^{-1}(L)$ of L ,¹¹
- a function $DimOperatorsAtLvl_D: Lvs_D \rightarrow \mathcal{P}(DimOperators_D)$ that maps each level $L \in Lvs_D$ to the set of dimensional operators $LvlOfDimOperator_D^{-1}(L)$ of L ,
- a function $DimPredicatesAtLvl_D: Lvs_D \rightarrow \mathcal{P}(DimPredicates_D)$ that maps each level $L \in Lvs_D$ to the set of dimensional predicates $LvlOfDimPredicate_D^{-1}(L)$ of L , and
- a function $JoinConditionsAtLvl_D: Lvs_D \rightarrow \mathcal{P}(JoinConditions_D)$ that maps each level $L \in Lvs_D$ to the set of join conditions $LvlOfJoinCondition_D^{-1}(L)$ of L . □

A dimension schema D can be identified by a name obtained by $NameOf(D)$. For instance, in Figure 2.3, name *Doctor* identifies the dimension schema for doctors. Subsequently, we use the name of a dimension schema equivalent to the dimension schema itself.

Each dimension node of dimension schema D belongs exactly to one level and all levels of D represent different granularities with respect to dimension nodes. Concerning these granularities, the set of levels of a dimension are considered as a definition of a hierarchy specified by the roll-up relation $RollupRel_D$ (also symbolized by the arrow-notation \rightarrow and \Rightarrow). Level top_D represents the dimension level with coarsest and $base_D$ the one with finest granularity. If the context of a dimension D is obvious, we also write top for top_D and $base$ for $base_D$. In examples, we allow to use name top (in *slant*

¹¹Symbol \mathcal{P} is used to denote a power set and superscript $^{-1}$ after a function f is used to denote the inverse function of f (written as f^{-1}). Thus, $\mathcal{P}(DescrAttrs_D)$ represents the power set of $DescrAttrs_D$ and $LvlOfDescrAttr_D^{-1}(L)$ the inverse image of L under function $LvlOfDescrAttr_D$.

font) instead of the unique name $NameOf(top_D)$. As a difference, the name of a base level has to be unique over all dimensions, for instance, the base level of dimension schema *Doctor* is named as *doctor*. Each dimension level can have additional descriptive attributes specified by function $LvlOfDescrAttr_D$.

In [34], dimension levels are denoted as dimension attributes and descriptive attributes as non-dimension attributes. The notion of a dimension in [34] can be associated with the base level in our definition. Furthermore, Golairelli et al. allow to have dimension levels that roll up to more than one level of the same dimension. This can be considered as a dimension with more than one hierarchy. In the definition above, exactly one hierarchy is defined for a dimension (constrained by the property of a strict total order of relation $RollupRel_D$). This constraint simplifies subsequent definitions for APMN4BI and, on the other side, it does not restrict the approach of APMN4BI. For instance, if there exists more than one doctor hierarchy, one can introduce more than one appropriate dimension schema like a dimension schema for doctors' regions or a dimension schema for doctors' medical sections.

Dimension schemas in Figure 2.3 are graphically embedded in blue ellipse-like shapes. Dimension levels are depicted as white rounded rectangles (labelled by the name of the dimension level) and connected with lines representing the roll-up relation. Descriptive attributes (grey labels) are not marked by white circles but they are also visually linked to levels. Numeric descriptive attributes that can also be used as base measures are prefixed by a specific symbol for base measures. In our graphical representation, top levels are omitted. As an example, dimension schema *Doctor* comprises levels $Lvls_{Doctor} = \{doctor, docDistrict, docProvince, top_{Doctor}\} = \{doctor, docDistrict, docProvince, top\}$ with hierarchy $doctor \rightarrow docDistrict, docDistrict \rightarrow docProvince, docProvince \rightarrow top_{Doctor}$ (or $docProvince \rightarrow top$), or in shorthand notation $doctor \rightarrow docDistrict \rightarrow docProvince \rightarrow top$. At level *doctor*, measures can be analyzed for particular doctors, at level *docDistrict*, measures are aggregated to doctors' districts, and at level *docProvince* to doctors' provinces. The top-most level top_{Doctor} (or named as *top*) comprises all doctors' provinces. Each dimension level describes dimension nodes at schema level, for instance, doctors as dimension nodes belong to dimension

level *doctor*. Descriptive attributes specify additional properties of dimension nodes of a particular dimension level. For example, descriptive attribute $docAge \in DescrAttrsAtLvl_{Doctor}(doctor)$ specifies the age of a doctor or $inhPerSqkmInDocDistr \in DescrAttrsAtLvl_{Doctor}(docDistrict)$ specifies the number of inhabitants per square kilometer of doctors' district.

Dimensional operators and dimensional predicates represent enrichments of a common DFM. They refer to a dimension level. Dimensional operators are depicted as ellipses linked to the dimension level they refer to. In Figure 2.3, dimensional operators *nextYear* and *prevYear* are connected to level *year*, dimensional operators *qrtOfNextYear* and *qrtOfPrevYear* are linked to level *quarter*, and *monthOfNextYear* and *monthOfPrevYear* to level *month*.

In the graphical notation (see Figure 2.3), dimensional predicates are depicted as rounded rectangles comprising two parts, one containing the name of a dimensional predicate and the other one containing its definition as a boolean expression. A black ellipse is used as pictogram for dimensional predicates (placed in front of its name).

The example of Figure 2.3 comprises ten dimensional predicates: *DocInRuralDistrict*, *DocInUrbanDistrict*, *OldDoctor*, *OldDocInRuralDistrict*, *OldDocInUrbanDistrict*, *InsInRuralDistrict*, *InsInUrbanDistrict*, *OldInsurant*, *OldInsInRuralDistrict*, and *OldInsInUrbanDistrict*. The dimensional predicates *DocInRuralDistrict*, *DocInUrbanDistrict*, *OldDoctor*, *InsInRuralDistrict*, *InsInUrbanDistrict*, and *OldInsurant* are defined by the expressions $inhPerSqkmInDocDistr < 400$, $inhPerSqkmInDocDistr \geq 400$, $docAge > 55$, $inhPerSqkmInInsDistr < 400$, $inhPerSqkmInInsDistr \geq 400$, and $insAge > 65$. Note, name *DocInRuralDistrict*, for example, is used equivalently to the identified dimensional predicate itself, meaning $NameOf(DocInRuralDistrict)$ is equal to *DocInRuralDistrict*. Moreover, $ExprOf(DocInRuralDistrict)$ obtains expression $inhPerSqkmInDocDistr < 400$. In the definition of dimensional predicates *OldDocInRuralDistrict*, *OldDocInUrbanDistrict*, *OldInsInRuralDistrict*, and *OldInsInUrbanDistrict*, names of other dimensional predicates are used. These dimensional predicates are defined by the expressions *OldDoctor* AND *DocInRuralDistrict*, *OldDoctor* AND *DocInUrbanDistrict*, *OldInsurant* AND *InsInRuralDistrict*, and *OldInsurant* AND *InsInUrban-*

District. Notice that, for instance, $ExprOf(OldDocInRuralDistrict)$ also resolves contained names of other dimensional predicates and thus is equal to expression $docAge > 55 \text{ AND } inhPerSqkmInDocDistr < 400$.¹²

We use implications (also denoted as subsumptions) for predicates to define hierarchies of dimensional predicates. Such hierarchies can be used by analysts for navigation. By navigation along such subsumption hierarchies, a business analyst can narrow or broaden restrictions for selecting dimension nodes.

Definition 2.5. If P_1 and P_2 are dimensional predicates and P_1 implies P_2 , we write $P_1 \Rightarrow P_2$ and also say P_2 *subsumes* P_1 . In this case, P_2 is the *subsumer* of P_1 and, reversely, P_1 is the *subsumee* of P_2 . \square

At instance level, the implication $P_1 \Rightarrow P_2$ of dimensional predicates P_1 and P_2 means that the set of dimension nodes which satisfy dimensional predicate P_1 is a subset of the set of dimension nodes which satisfy dimensional predicate P_2 . In an enriched DFM diagram, a subsumption relation is depicted as an arrow with a non-filled arrowhead (similar to generalization in UML). Figure 2.3 shows that predicate *OldDocInRuralDistrict* implies predicates *OldDoctor* and *DocInRuralDistrict*, or in other words, *OldDoctor* and *DocInRuralDistrict* subsume *OldDocInRuralDistrict*. Thus, in Figure 2.3, there are arrows from *OldDocInRuralDistrict* to *OldDoctor* and to *DocInRuralDistrict*. Similarly, dimension predicates *OldDoctor* and *DocInUrbanDistrict* subsume *OldDocInUrbanDistrict*, *OldInsurant* and *InsInRuralDistrict* subsume *OldInsInRuralDistrict*, and *OldInsurant* and *InsInUrbanDistrict* subsume *OldInsInUrbanDistrict*.

In this thesis, subsumption hierarchies are considered as given. Within the scope of the project *semCockpit*, predicates¹³ were mapped into OWL and by the usage of OWL reasoners, subsumption checking was performed automatically [91]. A detailed description of the mappings into OWL can be found in [95].

Dimensional predicates are connected to the dimension level they refer to

¹²If necessary, parentheses are used to guarantee correct order of evaluation.

¹³In *semCockpit*, the term *concept* is used instead of the term *predicate*.

(determined by function $LvlOfDimPredicate_D$). Graphically, this is depicted by a line from the dimensional predicate to the corresponding dimension level.¹⁴ This means that a dimensional predicate can be applied to all nodes of that level and also to all nodes of all sublevels. In the example of Figure 2.3, dimension predicates *DocInRuralDistrict* and *DocInUrbanDistrict* are connected to level *docDistrict*, *OldDoctor* to level *doctor*, *InsInRuralDistrict* and *InsInUrbanDistrict* to level *insDistrict*, and *OldInsurant* to level *insurant*. This means that dimensional predicates *DocInRuralDistrict* and *DocInUrbanDistrict* can be applied to dimension nodes of level *docDistrict* and level *doctor*, dimensional predicate *OldDoctor* can be only applied to dimension level *doctor*, dimensional predicates *InsInRuralDistrict* and *InsInUrbanDistrict* can be applied to dimension levels *insDistrict* and *insurant*, and *OldInsurant* can be only applied to dimension level *insurant*. Note that, for instance, a line between dimensional predicate *OldDocInRuralDistrict* and dimension level *doctor* does not need to be drawn because the corresponding dimension level *doctor* associated with this dimensional predicate results from its subsumers. In the present case, dimension level *doctor* associated with subsumer *OldDoctor* is also used as dimension level associated with dimensional predicate *OldDocInRuralDistrict* (as a subsumee). Generally speaking, if there is a line from a dimensional predicate to a dimension level (possibly not drawn in the case of subsumptions), the dimensional predicate can be applied to the nodes of this dimension level and also to all nodes of all dimensional sublevels where lines between these sublevels and this dimensional predicate do not need to be drawn.

The example of Figure 2.3 comprises two join conditions which are depicted as rounded rectangles, similar to dimensional predicates. They are identified by a unique name and connected by a line to the dimension level they refer to. Join condition with name *SameInsProvince* is linked to dimension level *insProvince* and is defined by boolean expression $CoI.insProvince = CoC.insProvince$. *PrevYear* is the name of a join condition defined by

¹⁴In the case of subsumptions, the line between the subsumee and the corresponding dimension level can be omitted because this dimension level can be determined by the subsumers.

expression $prevYear(\text{CoI}.year) = \text{CoC}.year$ that is linked to dimension level $year$.

Finally, in this subsection, we give a separate definition for dimension schema $Time$ which will be used later in this thesis. This dimension schema has $date$ as base level that rolls up to $month$, $quarter$, and $year$. Level top_{Time} represents the top level of time containing all years.

Definition 2.6. The dimension schema $Time$ is defined by $Lvls_{Time} = \{date, month, quarter, year, top_{Time}\} = \{date, month, quarter, year, top\}$, $RollupRel_{Time} = \{(date, month), (month, quarter), (quarter, year), (year, top)\}$, $DescrAttrs_{Time} = \emptyset$, $LvlOfDescrAttr_{Time} = \emptyset$, $DimOperators_{Time} = \{nextYear, prevYear, monthOfNextYear, monthOfPrevYear, qrtOfNextYear, qrtOfPrevYear\}$, $LvlOfDimOperator_{Time} = \{(nextYear, year), (prevYear, year), (qrtOfNextYear, quarter), (qrtOfPrevYear, quarter), (monthOfNextYear, month), (monthOfPrevYear, month)\}$, $DimPredicates_{Time} = \emptyset$, $LvlOfDimPredicate_{Time} = \emptyset$, $JoinConditions_{Time} = \{PrevYear\}$, and $LvlOfJoinCondition_{Time} = \{(PrevYear, year)\}$. \square

This definition corresponds to the example in Figure 2.3. In this example, no descriptive attributes and no dimensional predicates are defined for dimension schema $Time$. One can also consider other definitions comprising descriptive attributes as well as dimensional predicates.

2.4.1.2 Dimension Instances

A dimension instance defines dimension nodes of a dimension schema. The dimension nodes are partitioned accordingly to the level hierarchy. Additionally, there is a “sub-super-node-relation” such that each sub-node is associated unambiguously to its super-node. Moreover, each dimension node has property values required by descriptive attributes defined in the dimension schema.

Definition 2.7. A *dimension instance* $\mathbf{d} = (D, Nodes_{\mathbf{d}}, LvlOfNode_{\mathbf{d}}, NodeOrder_{\mathbf{d}}, SuperNodeOf_{\mathbf{d}}, DescrAttrVals_{\mathbf{d}})$ ¹⁵ of a dimension schema D with

¹⁵For referencing objects at instance level like dimension instances, dimension nodes, or cube instances, we use a **bold** font.

$Lvls_D = \{L_1, \dots, L_p\}$, $L_1 = base_D$, $L_p = top_D$, and $L_1 \rightarrow \dots \rightarrow L_p$ comprises

1. a finite set of dimension nodes $Nodes_{\mathbf{d}}$ containing at least two elements, one denoted as $\mathbf{all}_{\mathbf{d}}$ (*all node*),
2. a surjective function $LvlOfNode_{\mathbf{d}}: Nodes_{\mathbf{d}} \rightarrow Lvls_D$ such that $LvlOfNode_{\mathbf{d}}(\mathbf{all}_{\mathbf{d}}) = top_D$,
3. a set of strict linear orderings $NodeOrder_{\mathbf{d}} = \{\prec_1, \dots, \prec_{p-1}\}$ such that, for $1 \leq i \leq p-1$, \prec_i is a strict linear ordering on $NodesOfLvl_{\mathbf{d}}(L_i)$, where $NodesOfLvl_{\mathbf{d}}$ is defined as $NodesOfLvl_{\mathbf{d}}: Lvls_D \rightarrow \mathcal{P}(Nodes_{\mathbf{d}})$ such that, for $L \in Lvls_D$, $NodesOfLvl_{\mathbf{d}}(L) = LvlOfNode_{\mathbf{d}}^{-1}(L)$,
4. a set of surjective functions $SuperNodeOf_{\mathbf{d}} = \{SuperNodeOf_{\mathbf{d}}^1, \dots, SuperNodeOf_{\mathbf{d}}^{p-1}\}$ such that, for $1 \leq i \leq p-1$, $SuperNodeOf_{\mathbf{d}}^i: NodesOfLvl_{\mathbf{d}}(L_i) \rightarrow NodesOfLvl_{\mathbf{d}}(L_{i+1})$, and
5. a set of functions $DescrAttrVals_{\mathbf{d}} = \{DescrAttrVals_{\mathbf{d}}^1, \dots, DescrAttrVals_{\mathbf{d}}^{p-1}\}$ such that, for $1 \leq i \leq p-1$, $DescrAttrVals_{\mathbf{d}}^i: NodesOfLvl_{\mathbf{d}}(L_i) \rightarrow dom(A_{i,1}) \times \dots \times dom(A_{i,q_i})$ with $DescrAttrsAtLvl_D(L_i) = \{A_{i,1}, \dots, A_{i,q_i}\}$ and dom returning the domain of a descriptive attribute.

Moreover, we define the following: $BaseNodes_{\mathbf{d}} = NodesOfLvl_{\mathbf{d}}(L_1)$, $DimSchema_{\mathbf{d}} = D$, $Lvls_{\mathbf{d}} = Lvls_D$, $base_{\mathbf{d}} = base_D$, $top_{\mathbf{d}} = top_D$, $DescrAttrs_{\mathbf{d}} = DescrAttrs_D$, $DimOperators_{\mathbf{d}} = DimOperators_D$, $DimPredicates_{\mathbf{d}} = DimPredicates_D$, $JoinConditions_{\mathbf{d}} = JoinConditions_D$, $DimOperatorsAtLvl_{\mathbf{d}} = DimOperatorsAtLvl_D$, $DimPredicatesAtLvl_{\mathbf{d}} = DimPredicatesAtLvl_D$, $JoinConditionsAtLvl_{\mathbf{d}} = JoinConditionsAtLvl_D$, and, for $1 \leq i \leq i+k \leq p-1$, $SuperNodeOf_{L_i, L_{i+k+1}}^{\mathbf{d}} = SuperNodeOf_{i+k}^{\mathbf{d}} \circ \dots \circ SuperNodeOf_i^{\mathbf{d}}$.¹⁶

For $\mathbf{n}_1, \mathbf{n}_2 \in Nodes_{\mathbf{d}}$, \mathbf{n}_1 is a *direct subnode* of \mathbf{n}_2 (written as $\mathbf{n}_1 \rightarrow \mathbf{n}_2$), if there exists a function $SuperNodeOf_i^{\mathbf{d}} \in SuperNodeOf_{\mathbf{d}}$ such that $SuperNodeOf_i^{\mathbf{d}}(\mathbf{n}_1) = \mathbf{n}_2$. Recursively, for $\mathbf{n}_1, \mathbf{n}_2 \in Nodes_{\mathbf{d}}$, we define that \mathbf{n}_1 is

¹⁶Symbol \circ represents function composition.

a *subnode* of \mathbf{n}_2 (written as $\mathbf{n}_1 \twoheadrightarrow \mathbf{n}_2$), if $\mathbf{n}_1 \rightarrow \mathbf{n}_2$ or there exists an $\mathbf{n} \in \text{Nodes}_{\mathbf{d}}$ such that $\mathbf{n}_1 \twoheadrightarrow \mathbf{n}$ and $\mathbf{n} \twoheadrightarrow \mathbf{n}_2$. Finally, we define DimInstances_D to be the set of all dimension instances of dimension schema D . \square

In the rest of this thesis, we also use the term dimension as a synonym for a dimension instance and dimension nodes are also denoted as nodes. As for dimension schemas, a dimension instance \mathbf{d} is identified by a unique name obtained by $\text{NameOf}(\mathbf{d})$. Similarly to dimension schemas, we use $\text{NameOf}(\mathbf{d})$ equivalent to the dimension instance \mathbf{d} itself.

A dimension instance \mathbf{d} of dimension schema D consists of a set of nodes $\text{Nodes}_{\mathbf{d}}$. $\text{NodesOfLvl}_{\mathbf{d}}(L)$ denotes the set of nodes of level L and each node is mapped exactly to one level (defined by function $\text{LvlOfNode}_{\mathbf{d}}$). $\text{BaseNodes}_{\mathbf{d}}$ are the nodes of the base level of finest granularity and $\mathbf{all}_{\mathbf{d}}$ is the unique node at level top_D with coarsest granularity that comprises all other nodes as sub-nodes. Set $\text{SuperNodeOf}_{\mathbf{d}}$ defines the sub-super-node relation. If the context of the dimension is obvious, we also write \mathbf{all} for $\mathbf{all}_{\mathbf{d}}$. Except $\mathbf{all}_{\mathbf{d}}$, each node belongs to exactly one super-node. To offer navigation operators moving to the previous or to the next dimension node, we introduce a strict linear order relation \prec_i per level L_i (except for the level top_D). As default order relation we take alphabetical ordering on node names except for dimension instances of dimension schema *Time* where we assume to have a temporal order relation. The functions of set $\text{DescrAttrVals}_{\mathbf{d}}$ return values for all describing attributes of all dimension nodes.

Figure 2.4 and Figure 2.5 visualize an excerpt of a cube instance that contains dimension instances **Time** and **Doctor** of dimension schemas *Time* and *Doctor* of Figure 2.3.¹⁷ In our examples, we only demonstrate one instance per schema. Thus we use same names at schema and instance level. Although we refrain from introducing separate names for dimension instances, for example, one can consider that these instances are obtained from a particular insurer at a particular point of time.

¹⁷We use different fonts to distinguish between instance and schema level. Dimension instances are written in a **sans serif** font and dimension schemas in a *slant* font. In graphical representations (see, for instance, Figure 2.4 and Figure 2.5), again different font size and thickness is used to distinguish names for different items (similar to schema level).

In a consistent manner, also a dimension node \mathbf{n} is identified by a name obtained by $NameOf(\mathbf{n})$. Again this name is used equivalently to the node itself. Thus, for instance, $NameOf(\text{Dr. No})$ is equivalent to Dr. No .¹⁸ We also introduce the notion of an expression of a dimension node \mathbf{n} which is obtained by $ExprOf(\mathbf{n})$. Such a dimension node expression can be interpreted as a literal satisfying a specific syntax. For example, the expression returned by $ExprOf(\text{Dr. No})$ could be syntactically defined as `'Dr. No'` which can be used in an SQL like syntax. In the case of node \mathbf{all}_d , we refrain from using a unique name per dimension instance \mathbf{d} and use name \mathbf{all} ¹⁹ uniformly for all dimension instances.

Note that a dimension schema is a part of the definition of a dimension instance. If there are two dimension instances \mathbf{d}_1 and \mathbf{d}_2 of dimension schemas D_1 and D_2 , respectively, and $D_1 \neq D_2$, then also $\mathbf{d}_1 \neq \mathbf{d}_2$, even if $Nodes_{\mathbf{d}_1} = Nodes_{\mathbf{d}_2}$, $LvlOfNode_{\mathbf{d}_1} = LvlOfNode_{\mathbf{d}_2}$, $NodeOrder_{\mathbf{d}_1} = NodeOrder_{\mathbf{d}_2}$, and $SuperNodeOf_{\mathbf{d}_1} = SuperNodeOf_{\mathbf{d}_2}$, and $DescrAttrVals_{\mathbf{d}_1} = DescrAttrVals_{\mathbf{d}_2}$. This would be the case, if dimension schemas D_1 and D_2 only differ in dimensional operators, dimensional predicates, or join conditions—all of them are components that only exists at schema level and that can be applied to dimension instances.

Figure 2.4 contains an abstract visualization of a cube example where exemplary dimension nodes and some subnode relations are depicted. In Figure 2.5, there is a sketch of a possible relational implementation in a star schema. One can see base nodes Dr. No and Dr. Marbuse in dimension instance Doctor of dimension schema Doctor at level doctor . Both dimension nodes have a common unique super-node Linz-Stadt at level docDistrict (i.e., $\text{Dr. No} \rightarrow \text{Linz-Stadt}$ and $\text{Dr. Marbuse} \rightarrow \text{Linz-Stadt}$) that itself rolls up to super-node Upper Austria at level docProvince (i.e., $\text{Linz-Stadt} \rightarrow \text{Upper Austria}$). Finally, all provinces roll up to the unique node $\mathbf{all}_{\text{Doctor}}$ which can be identified by a unique name, for instance by name $\mathbf{all\ doctors}$. For simplicity, if there are no ambiguities, we also use name \mathbf{all} for node \mathbf{all}_d of

¹⁸We use a sans serif font for dimension nodes (e.g., Dr. No) and abstract from implementation specific notations (e.g., `'Dr. No'` enclosed by single quotation marks) for constant names depending on data types like string, number, or date.

¹⁹Again written in a sans serif font.

every used dimension instance \mathbf{d} .²⁰ Therefore, the assertion that all provinces roll up to the unique node $\mathbf{all}_{\text{Doctor}}$ can be written as $\text{Upper Austria} \rightarrow \mathbf{all}$.

Moreover, we have a function per level (except for the top level) that maps a node to values for its descriptive attributes defined in the dimension schema. In the relational implementation of Figure 2.5, descriptive attributes and their values are depicted for dimension **Doctor**. For instance, node **Dr. No** of level *doctor* has one descriptive attribute *docAge*. There is a function $\text{DescrAttrValsOfDoctor} \in \text{DescrAttrVals}_{\text{Doctor}}$ such that $\text{DescrAttrValsOfDoctor}: \text{Nodes}_{\text{Doctor}}(\text{doctor}) \rightarrow \text{dom}(\text{docAge})$. In the presented example, we have $\text{DescrAttrValsOfDoctor}(\text{Dr. No}) = 56$. Similarly, there is a function $\text{DescrAttrValsOfDocDistrict} \in \text{DescrAttrVals}_{\text{Doctor}}$, for descriptive attribute *inhPerSqkmInDocDistr*, such that $\text{DescrAttrValsOfDocDistrict}: \text{Nodes}_{\text{Doctor}}(\text{docDistrict}) \rightarrow \text{dom}(\text{inhPerSqkmInDocDistr})$ where, for example, $\text{DescrAttrValsOfDocDistrict}(\text{Linz-Stadt}) = 2115$. In these examples, both functions return a single value. If a dimension level comprises more than one descriptive attribute, the corresponding function would return a tuple of values.

Furthermore, there is a strict linear order relation on the nodes of a dimension level. For example, let \prec_{doctor} , $\prec_{\text{docDistrict}}$, and $\prec_{\text{docProvince}} \in \text{NodeOrder}_{\text{Doctor}}$ be strict linear order relations for levels *doctor*, *docDistrict*, and *docProvince* of dimension **Doctor**, all representing alphabetical ordering. In this case, we have, for instance, **Dr. Marbuse** \prec_{doctor} **Dr. No** at level *doctor*, **Kirchdorf** $\prec_{\text{docDistrict}}$ **Linz-Land** at level *docDistrict*, and **Lower Austria** $\prec_{\text{docProvince}}$ **Upper Austria** at level *docProvince*. In dimension **Time**, we use temporal ordering of the dimension nodes of a dimension level (denoted as \prec_{year} , \prec_{quarter} , \prec_{month} , $\prec_{\text{date}} \in \text{NodeOrder}_{\text{Time}}$). For example, we have **2015** \prec_{year} **2016** at level *year*, **2015Q1** \prec_{quarter} **2015Q2** at level *quarter*, **May 2015** \prec_{month} **June 2015** \prec_{month} **July 2015** \prec_{month} **August 2015** at level *month*, or **1 May 2015** \prec_{date} **2 May 2015** at level *date*. If the context is clear, we use the symbol \prec for all such order relations. Thus, for example, we simply write **Dr. Marbuse** \prec **Dr. No** and **1 May 2015** \prec **2 May 2015** instead of **Dr. Marbuse** \prec_{doctor} **Dr. No** and **1 May 2015** \prec_{date} **2 May 2015**.

²⁰Name \mathbf{all} for node $\mathbf{all}_{\mathbf{d}}$ of a dimension instance \mathbf{d} is written again in a sans serif font.

Dimensional operators are interpreted at instance level as functions that map a dimension node of a certain dimension level to a value of a certain domain that again could comprise dimension nodes of the same dimension level—in this case, a dimensional operator expresses a dimension node by another node of the same level. Dimensional operators *nextYear* and *prevYear* of Figure 2.3 map a year to the next or previous year. For example, *nextYear*(2016) expresses year 2017 and *prevYear*(2016) indicates year 2015. *qrtOfNextYear* and *qrtOfPrevYear* are dimensional operators at level *quarter* that map a quarter of a year to the same quarter of the next or previous year. As an example, *qrtOfNextYear*(2016Q1) corresponds to node 2017Q1 and *qrtOfPrevYear*(2016Q1) corresponds to node 2015Q1. Similarly, dimensional operators *monthOfNextYear* and *monthOfPrevYear*, associated to level *month*, map a month of a year to the same month of the next or previous year. For instance, *monthOfNextYear*(201607) represents dimension node 201707 and *qrtOfPrevYear*(201607) corresponds to node 201507.

Moreover, we assume to have dimensional operators for nodes of every dimension level that return the next and previous node with respect to order relation \prec .²¹ As an example, we assume that dimensional operators *nextDoctor* and *prevDoctor* are defined at dimension level *doctor* and they return the next or previous doctor with respect to alphabetical ordering. For instance, *nextDoctor*(Dr. Marbuse) returns node Dr. No in the case that node Dr. No follows directly node Dr. Marbuse in the alphabetical order of the corresponding dimension instance.

At instance level, dimensional predicates are 1-ary predicates defined over dimension nodes of the dimension level associated with the dimensional predicate and over dimension nodes of all sublevels of the associated dimension level. For instance, in the example of Figure 2.3, dimensional predicate *DocInUrbanDistrict* can be applied to dimension nodes of level *docDistrict* and to dimension nodes of level *doctor*. For a dimension node, the predicate is true, if the defining expression of the dimensional predicate evaluates to true for this dimension node. As depicted in Figure 2.3, dimensional predicate

²¹These operators are not depicted in Figure 2.3, except dimensional operators *nextYear* and *prevYear* at dimension level *year*.

DocInUrbanDistrict can be applied to dimension node *Linz-Stadt* (shown in Figure 2.4 and Figure 2.5, respectively) of level *docDistrict*. The evaluation of expression *inhPerSqkmInDocDistr* \geq 400 returns true because descriptive attribute *inhPerSqkmInDocDistr* of dimension node *Linz-Land* contains value 2115 (see column *inhPerSqkmInDocDistr* of dimension table *Doctor* in Figure 2.5) which is greater than 400. Dimensional predicate *DocInUrbanDistrict* can also be applied to dimension node *Dr. No* of level *doctor* which is a sublevel of level *docDistrict*. In this case, the defining expression also evaluates to true because *Dr. No* is a subnode of dimension node *Linz-Stadt* with value 2115 for descriptive attribute *inhPerSqkmInDocDistr*.

A join condition is used to combine records of two result sets (context of interest and context of comparison) with respect to nodes of the join condition's dimension level. As mentioned in the previous Subsection 2.4.1.1, join conditions provide a prerequisite for comparisons which are represented in APMN4BI as comparative analysis situations that will be introduced in Section 3.2 of Chapter 3. Subqueries induced by the context of interest and the context of comparison are linked by a join condition.²² The dimension levels used in a join condition are qualified by *CoI* and *CoC* to refer to the context of interest and the context of comparison, respectively. Records of a result set of the context of interest and records of a result set of the context of comparison are joined, if the corresponding dimension nodes satisfy the join condition. In the case of join condition *SameInsProvince*, records are combined, if boolean expression *CoI.insProvince* = *CoC.insProvince* becomes true for dimension nodes at level *insProvince*. For example, *Upper Austria* in the context of interest is joined with *Upper Austria* in the context of comparison. Regarding join condition *PrevYear*, records are joined, if boolean expression *PrevYear*(*CoI.year*) = *CoC.year* becomes true for dimension nodes at level *year*. In this example, dimensional operator *PrevYear* is applied to dimension nodes in the context of interest. For instance, year 2018 in the context of interest is joined with previous year 2017 in the context of comparison.

²²With respect to translation into SQL, a join condition represents a part of an SQL-on-or SQL-where-clause.

2.4.2 Measures and Measure Predicates

Before defining cube schemas and cube instances, various types of measures and measure predicates are introduced. Measures are divided into base measures and aggregate measures. Base measures comprise simple base measures, numeric descriptive attributes, and derived base measures as subtypes. Simple aggregate measures and derived aggregate measures are subtypes of aggregate measures. Finally, scores can be considered as measures used for comparing aggregate measures of two queries.

Additionally, predicates founded on base measures, aggregate measures, and scores are introduced. Base measure predicates are defined to restrict the selection of a cube's facts. In contrast, aggregate measure predicates and score predicates are used to filter records of a query result set.

2.4.2.1 Base Measures

A base measure describes a numeric value which is associated with exactly one fact. Simple base measures, numeric descriptive attributes, and derived base measures are considered as subtypes of base measures. As explained in the introduction of Section 2.4, simple base measures are given elements of a specific universe. At this point, we additionally claim that at instance level a simple base measure represents a numeric domain. The quantity and the costs of drug prescriptions or ambulant treatments, or the number of days and the costs of hospitalizations are examples of simple base measures. Analogously to simple base measures, descriptive attributes are also considered as elements of a specific universe (see introduction of Section 2.4). In general, we allow that descriptive attributes can belong to a numeric or to a non-numeric domain. The first name or the last name of an insurant are examples of non-numeric descriptive attributes. Otherwise, the age of an insurant or the number of inhabitants of an insurants' district are examples of numeric descriptive attributes. In the context of base measures, we have to restrict to numeric descriptive attributes as subtype. The subsequent definition specifies derived base measures as arithmetic expressions that, of course, also belong to numeric domains. Simple base measures, numeric de-

scriptive attributes, and derived base measures belong to numeric domains and represent subtypes of base measures that, as a consequence, also belong to numeric domains.

Definition 2.8. A *derived base measure* is an arithmetic expression defined over a set of simple base measures, numeric descriptive attributes, and numeric constants. \square

Although we do not define the syntax of arithmetic expressions of derived base measures, one can assumed SQL based notations. Arithmetic operators like $+$, $-$, $*$, $/$, MOD or % (modulo operator), and parentheses (and) are used in combination with simple base measures, numeric descriptive attributes, and numeric constants.

Definition 2.9. A *base measure* is a simple base measure, a numeric descriptive attribute, or a derived base measure. A set of base measures $BMsr$ is *closed*, if it only contains derived base measures that are defined over elements of $BMsr$ itself and numeric constants. \square

In Figure 2.3, *quantity* and *costs* are two simple base measures used in drug prescriptions and ambulant treatments to indicate the quantity and actual total costs of prescribed drugs and ambulant treatments. For hospitalizations, simple base measures *days* and *costs* are used representing the number of days and the actual total costs of a hospital stay.

Numeric descriptive attributes *drugPrice*, *medServFee*, *docAge*, *insAge*, *inhPerSqkmInDocDistr*, *inhPerSqkmInHospDistr*, and *inhPerSqkmInInsDistr* can be considered as a type of base measures that describe dimension nodes of a certain dimension level. *drugPrice* and *medServFee* specify the list price of drugs and the fee of medical services. The age of doctors and insurants is stated by *docAge* and *insAge*. Doctors, hospitals, and insurants are located in districts which are described additionally by attributes *inhPerSqkmInDocDistr*, *inhPerSqkmInHospDistr*, and *inhPerSqkmInInsDistr*.

Derived base measures are considered as arithmetic expressions over simple base measures, numeric descriptive attributes, and numeric constants. The arithmetic expression *costs / quantity* comprises simple base measures

costs and *quantity*. It represents a derived base measure associating the costs per drug unit or the costs per unit of a medical service. The arithmetic expression $quantity * drugPrice * 1.1$ is another derived base measure that consists of simple base measure *quantity*, numeric descriptive attribute *drugPrice*, and numeric constant 1.1. It can be interpreted as drug prescription costs assessed by list prices and increased by taxes of ten percent.

Simple base measures like *quantity*, *costs*, and *days*, and descriptive attributes like *drugPrice*, *medServFee*, *docAge*, *insAge*, *inhPerSqkmInDocDistr*, *inhPerSqkmInHospDistr*, and *inhPerSqkmInInsDistr* are named elements of two corresponding universes (the universe of simple base measures and the universe of descriptive attributes).

Derived base measures are arithmetic expressions that can also be named. The name of a derived base measure b is obtained by $NameOf(b)$ which is used equivalently to the derived base measure itself. The arithmetic expression of a derived base measure b is expressed by $ExprOf(b)$ such that names of other derived base measures are resolved recursively.

In the graphical representation, the name of a base measure is preceded by a symbol (a little square including a number). Derived base measures are represented by specific rectangles which are divided into two parts. The top part contains the name of the derived base measure (including the base measure symbol) and the bottom part contains its arithmetic expression. Such a rectangle is connected to a cube by a line, if the definition of the derived measure can be applied to the cube. If a derived base measure is used in the definition of another derived base measure, an arrow is drawn from the derived base measure that is used in the definition to the derived base measure that uses the other one. In this case, the line from the derived base measure, which represents the arrow's target, to the cube can be omitted.

2.4.2.2 Base Measure Predicates

Whereas dimensional predicates are used to narrow a set of dimension nodes, base measure predicates can be used to narrow a set of facts of a cube instance.

Definition 2.10. A *base measure predicate* is a boolean expression defined over a set of base measures. \square

Similar to dimensional predicates, we assume to have SQL based notations for boolean expressions that define base measure predicates. Again, such expressions can be considered as a part of an SQL-where-clause. In the example of Figure 2.3, base measure *costsPerUnit* is used to define base measure predicate *HighCostsPerUnit*. It is defined by the expression *costsPerUnit* > 50. As for dimensional predicates, base measure predicates are depicted as a rounded rectangle divided into two parts where the upper part contains the name of the base measure predicate and the lower part contains the defining boolean expression. The name of a base measure predicate can be used to refer to the base measure predicate, for instance, in the definition of other base measure predicates. Again, by function *NameOf* we obtain the name of a base measure predicate and the application of function *ExprOf* represents the boolean expression of a base measure predicate where names of other base measure predicates and names of derived base measures are resolved. The pictogram of a base measure predicate is drawn as a black ellipse containing a number (that symbolizes a predicate on base measures).

Note, numeric descriptive attributes can be used for both in definitions of dimensional predicates and in definitions of base measure predicates. It depends on the intention what an analyst wants to select. Descriptive attributes can be used in the definition of dimensional predicates to select dimension nodes independently from facts. On the other side, numeric descriptive attributes can be used in the definition of base measure predicates to select facts comprising dimension nodes and base measure values.

Although not depicted in Figure 2.3, implications (also denoted as subsumptions) of predicates are used to define hierarchies of base measure predicates (similar to subsumption hierarchies of dimensional predicates). Again, such hierarchies can be used by analysts for navigation, i.e., along such subsumption hierarchies, a business analyst can narrow or broaden restrictions for selecting facts.

Definition 2.11. If B_1 and B_2 are base measure predicates and B_1 implies

B_2 , we write $B_1 \Rightarrow B_2$ and also say B_2 *subsumes* B_1 . In this case, B_2 is the *subsumer* of B_1 and, reversely, B_1 is the *subsumee* of B_2 . \square

The implication $B_1 \Rightarrow B_2$ of base measure predicates B_1 and B_2 means that the set of facts which satisfy base measure predicate B_1 is a subset of the set of facts which satisfy base measure predicate B_2 . As for dimensional predicates, the subsumption relation for base measure predicates is depicted as an arrow with a non-filled arrowhead from the subsumee to the subsumer. Base measure predicates are connected to the graphical elements containing the base measures used for the definition of the base measure predicate. In the example of Figure 2.3, the definition of base measure predicate *HighCostsPerUnit* contains base measure *costsPerUnit*. Thus, a line from base measure predicate *HighCostsPerUnit* to base measure *costsPerUnit* is drawn. In the case of a subsumption, a line from the subsumee to a base measure that is used in the subsumee does not need to be drawn, if this base measure is also a part of the subsumer (analogously to subsumptions of dimensional predicates). A base measure predicate can be applied to cubes, if all base measures of the definition of the predicate are parts of that cube.

At instance level, base measure predicates are 1-ary predicates defined over facts. For a fact, the base measure predicate is true, if the defining expression of the base measure predicate evaluates to true for this fact. As depicted in Figure 2.3, base measure predicate *HighCostsPerUnit* can be applied to cubes *DrugPrescription*, *AmbTreatment*, and *Hospitalization*. Figure 2.4 (in an abstract visualization) and Figure 2.5 (in a relational representation) show an example of drug prescriptions at instance level. For the first visible fact in this depiction, costs amount to 100, i.e., costs of this fact are greater than 50. Thus base measure predicate *HighCostsPerUnit* is true for this fact. The next three facts exhibit costs not greater than 50. Thus, for these facts, base measure predicate *HighCostsPerUnit* evaluates to false.

2.4.2.3 Aggregate Measures

Whereas base measures are associated with single facts, aggregate measures accumulate facts and compute an aggregated value over these facts on the

basis of base measures and dimension nodes. For example, quantity and costs of drug prescriptions can be added up, insurants can be counted, and average costs of drug prescriptions can be calculated per unit or per insurant. In the subsequent definitions we specify simple aggregate measures and derived aggregate measures that are subtypes of aggregate measures.

Definition 2.12. A *simple aggregate measure expression* represents an expression that comprises an aggregation operator and an operand enclosed by parentheses, and possibly preceded by either a distinct- or an all-modifier. A *count operator* represents a specific subtype of aggregation operators. Base measures are used as operands. In the case of count operators, dimension levels and non-numeric descriptive attributes are also allowed as operands. Finally, a *simple aggregate measure* is defined as a simple aggregate measure expression. \square

Although the above definition does not specify an exact syntax of expressions formed by aggregation operators, we use SQL aggregation operators and the corresponding SQL syntax and SQL semantics as a prerequisite. The SQL operators `SUM`, `AVG`, `MIN`, `MAX`, and `STDDEV` compute the sum, the average, the minimum, the maximum, and the standard deviation of base measure values of facts. For counting, SQL operator `COUNT` can be used. SQL also provides modifiers `ALL` and `DISTINCT` as all- and distinct-modifiers, respectively. Modifier `ALL` has the effect that operand values of all records of a query result set are taken into account for aggregation (i.e., the same value can occur more than once for aggregation), whereas, modifier `DISTINCT` only allows to aggregate distinct operand values (i.e., if the same value occurs in more than one record in the result set, it will be still only taken into account once for aggregation). In SQL syntax, modifier `ALL` is considered as default and thus, in most cases, it is omitted.

In Figure 2.3, expressions `SUM(quantity)`, `SUM(costs)`, `SUM(days)`, and `COUNT(DISTINCT insurant)` are examples for simple aggregate measures.²³ The first three expressions are defined over base measures *quantity*, *costs*,

²³Note that one also could use modifier `ALL` explicitly and write `SUM(ALL quantity)`, `SUM(ALL costs)`, and `SUM(ALL days)`.

and *days*, and contain SQL operator `SUM`. They sum quantity and costs of drug prescriptions and ambulant treatments, and costs and days of hospitalizations. The last simple aggregate measure counts distinct dimension nodes of dimension level *insurant*. In this case, one only counts distinct insurants. If one would like to count all occurrences of an insurant, SQL expression `COUNT(ALL insurant)` (or, simply, `COUNT(insurant)`) has to be used.²⁴

Simple aggregate measures obtain names (again returned by function *NameOf*) that are used in the specification of derived aggregate measures. In the example of Figure 2.3, *SumOfQuantity*, *SumOfCosts*, *SumOfDays*, and *NumOfInsurants* are names for the simple aggregate measures `SUM(quantity)`, `SUM(costs)`, `SUM(days)`, and `COUNT(DISTINCT insurant)`. The expression of a simple aggregate measure is returned by function *ExprOf* such that contained names of derived base measures are also resolved.

Definition 2.13. A *derived aggregate measure* is an arithmetic expression defined over simple aggregate measures and numeric constants. \square

As for derived base measures, the syntax of arithmetic expressions of derived aggregate measures is defined in accordance with SQL. Again we use arithmetic operators `+`, `-`, `*`, `/`, `MOD` or `%` (modulo operator), and parentheses (and) in combination with simple aggregate measures and numeric constants. Derived aggregate measures obtain names by *NameOf* function and return the recursively resolved expression by function *ExprOf*.

Definition 2.14. An *aggregate measure* is either a simple or a derived aggregate measure. A set of aggregate measures *AMsr* is *closed*, if it only contains derived aggregate measures that are defined over elements of *AMsr* itself and numeric constants. \square

Analogously to simple aggregate measures, also derived aggregate measures are named. A name of a derived aggregate measure can be used

²⁴In SQL, also expression `COUNT(*)` is commonly used for counting all records of a result set also including null-values. In our approach of an eDFM and in the implementation as a star schema (see Subsection 2.4.4), we do not allow null-values, neither as dimension nodes nor as values for descriptive attributes, that also represents a common approach in dimensional modeling. Thus, in this thesis, SQL expression `COUNT(*)` is not used as a simple aggregate measure expression.

again in other derived aggregate measures. In the example of Figure 2.3, derived aggregate measures $SumOfCosts / SumOfQuantity$, $SumOfCosts / NumOfInsurants$, and $SumOfCosts / SumOfDays$ are identified by the names $AvgCostsPerUnit$, $AvgCostsPerInsurant$, and $AvgCostsPerDay$. Derived aggregate measure $AvgCostsPerUnit$ represents the average costs per unit and can be applied to drug prescriptions and ambulant treatments, whereas $AvgCostsPerDay$ represents the average costs per day and can only be applied to hospitalizations. $AvgCostsPerInsurant$ is a derived aggregate measure representing the average costs per insurant and which can be applied to drug prescriptions and ambulant treatments as well as to hospitalizations.

Simple aggregate measures and derived aggregate measures are specific subtypes of aggregate measures. In Figure 2.3, aggregate measures are drawn as rectangles containing two parts. The upper part comprises the name of an aggregate measure which is prefixed by a small multi-layer square including a number and the lower part contains the definition of an aggregate measure. Note, that the pictogram of an aggregate measure is depicted as a multi-layer square (symbolizing the aggregation of base measure values), whereas the pictogram of a base measure is drawn as a single-layer square.

If a simple aggregate measure can be applied to a cube, it is linked by a line to the graphical element containing the corresponding base measure or to the corresponding dimension level that is used for aggregation. For example, $SumOfQuantity$ is linked to rectangles $DrugPrescription$ and $AmbTreatment$ both containing base measure $quantity$, $SumOfCosts$ is linked to rectangles $DrugPrescription$, $AmbTreatment$, and $Hospitalization$ all containing base measure $costs$, $SumOfDays$ is linked to rectangle $Hospitalization$ containing base measure $days$, and $NumOfInsurants$ is linked to level $insurant$ of dimension $Insurant$. If a simple aggregate measure uses derived base measure $costsPerUnit$ or a numeric descriptive attribute like $inhPerSqkmInInsDistr$, it would be linked to rectangle $costsPerUnit$ or to dimension level $insDistrict$, respectively.

In the definition of derived aggregate measures, simple aggregate measures and other derived aggregate measures can be used. An arrow is drawn from an aggregate measure (simple aggregate measure as well as derived ag-

gregate measure) to a derived aggregate measure, if the first one is used in the definition of the second one. In the example of Figure 2.3, there are arrows from *SumOfQuantity* and from *SumOfCosts* to *AvgCostsPerUnit* indicating that simple aggregate measures *SumOfQuantity* and *SumOfCosts* are used in the definition of derived aggregate measure *AvgCostsPerUnit*. Similarly, there are arrows from *SumOfDays* and *SumOfCosts* to *AvgCostsPerDay* indicating that both simple aggregate measures are used in the definition of derived aggregate measure *AvgCostsPerDay*. Finally, one can see that the definition of derived aggregate measure *AvgCostsPerInsurant* comprises aggregate measures *NumOfInsurants* and *SumOfCosts* that is visualized by two arrows from *NumOfInsurants* and *SumOfCosts* to *AvgCostsPerInsurant*.

The definition of derived aggregate measures gives rise to the concept of aggregate measure hierarchies. An aggregate measure hierarchy can be used by a business analyst to navigate along it and to analyze dependent aggregate measures. To introduce aggregate measure hierarchies, we define a sub-aggregate-measure relationship.

Definition 2.15. If M_1 is an aggregate measure that is used as an operand in the arithmetic expression that defines another aggregate measure M_2 , then M_1 is a *direct sub-aggregate-measure* of M_2 (written as $M_1 \rightarrow M_2$). M_1 is a *sub-aggregate-measure* of M_2 (written as $M_1 \rightsquigarrow M_2$), if $M_1 \rightarrow M_2$, or if there exists an aggregate measure M , such that $M_1 \rightarrow M$ and $M \rightarrow M_2$.²⁵ \square

In Figure 2.3, a sub-aggregate-measure relation is depicted as an arrow from the sub-aggregate-measure to the super-aggregate-measure. There are the following examples of direct sub-aggregate-measures: $SumOfQuantity \rightarrow AvgCostsPerUnit$, $SumOfCosts \rightarrow AvgCostsPerUnit$, $NumOfInsurants \rightarrow AvgCostsPerInsurant$, $SumOfCosts \rightarrow AvgCostsPerInsurant$, $SumOfDays \rightarrow AvgCostsPerDay$, $SumOfCosts \rightarrow AvgCostsPerDay$.

²⁵In this definition, we require that sub-aggregate-measures are modeled explicitly for the usage in a sub-aggregate-measure relation. We do not presuppose general expressions as used, for instance, in the software system *Mathematica*.

2.4.2.4 Aggregate Measure Predicates

In contrast to dimensional predicates and base measures predicates, aggregate measure predicates do not restrict the set of dimension nodes or facts but filter the result set of queries.

Definition 2.16. An *aggregate measure predicate* is a boolean expression defined over a set of aggregate measures. \square

We assume to have SQL based notations for boolean expressions that define aggregate measure predicates. Such expressions can be considered as a part of an SQL-having-clause. Figure 2.3 shows an aggregate measure predicate defined by the expression $AvgCostsPerInsurant > 1000$. An aggregate measure predicate can also be named (in Figure 2.3 name *HighAvgDrugPrescrCostsPerIns* is used) and, graphically, it is also depicted as a rounded rectangle containing the name (prefixed with a filter symbol as a pictogram associating the filtering of a query result set) in the upper part and the defining expression in the lower part.²⁶

As for dimensional predicates and base measure predicates, also for aggregate measure predicates, implications are used to define subsumption hierarchies (not depicted in Figure 2.3). Such hierarchies can be used by analysts to narrow or broaden restrictions on query result sets.

Definition 2.17. If F_1 and F_2 are aggregate measure predicates and F_1 implies F_2 , we write $F_1 \Rightarrow F_2$ and also say F_2 *subsumes* F_1 . In this case, F_2 is the *subsumer* of F_1 and, reversely, F_1 is the *subsumee* of F_2 . \square

Subsumption relations for aggregate measure predicates are depicted as arrows with a non-filled arrowhead. Aggregate measure predicates are connected to aggregate measures that are used for the predicate definition. In Figure 2.3, the definition of aggregate measure predicate *HighAvgDrugPrescrCostsPerIns* contains aggregate measure *AvgCostsPerInsurant*. Thus a line from aggregate measure predicate *HighAvgDrugPrescrCostsPerIns* to

²⁶Again, the name and the recursively resolved expression of an aggregated measure predicate can be obtained by *NameOf* and *ExprOf*.

aggregate measure *AvgCostsPerInsurant* is drawn. If there is a subsumption arrow from one to another aggregate measure predicate, a line from the source predicate to the aggregate measure (used for defining the aggregate measure predicate) is omitted. An aggregate measure predicate can be applied to cubes, if all aggregate measures of the definition of the predicate are parts of that cube. As depicted in Figure 2.3, aggregate measure predicate *HighAvgDrugPrescrCostsPerIns* can be applied to cubes *DrugPrescription*, *AmbTreatment*, and *Hospitalization*.

At instance level, aggregate measure predicates are 1-ary predicates defined over the records of a result set of a query. For a record of a result set, the aggregate measure predicate is true, if the defining expression of the aggregate measure predicate evaluates to true for this record. If, for a record of the query result set, the value of aggregate measure *AvgCostsPerInsurant* is greater than 1000, the record is filtered, else it will be omitted.

2.4.2.5 Scores

Scores can be considered as special measures that are used to compare measure values of two related but different queries. One can think of two result sets which are compared by scores such that one result set receives the focus of interest (*context of interest*) and the second one is used for comparison (*context of comparison*).

Definition 2.18. A *score* is an arithmetic expression defined over qualified aggregate measures qualified by CoI and CoC, and numeric constants. \square

The definition of a score comprises aggregate measures. Each of them either refer to the context of interest or to the context of comparison. If the name of an aggregate measure is prefixed by CoI with a dot as separation, the aggregate measure refers to the context of interest. On the other side, prefix CoC denotes that an aggregate measure belongs to the context of comparison.

As for measures, also scores obtain names that can be used elsewhere to refer to the score.²⁷ Scores are visualized similarly to measures by rectangles

²⁷Again, the name and the recursively resolved expression of a score can be obtained by *NameOf* and *ExprOf*.

comprising two parts. The upper part contains the name of a score and the lower part its definition. Score visualization only has a different pictogram compared to the visualization of aggregate measures. Instead of the symbol for aggregate measures, scores are symbolized by a pictogram containing two concentric circles with different diameters denoting the difference between the context of interest and the context of comparison. If an aggregate measure is used in the definition of a score, there is an arrow from this aggregate measure to the score (analogously to arrows between two aggregate measures).

In the example of Figure 2.3, two scores with names *RatioOfSumOfCosts* and *RatioOfAvgCostsPerInsurant* are defined. Score *RatioOfSumOfCosts* is defined by the arithmetic expression $\text{CoI} . \text{SumOfCosts} / \text{CoC} . \text{SumOfCosts}$. It calculates the ratio of the sum of costs of the context of interest and the sum of costs of the context of comparison. Because aggregate measure *SumOfCosts* is used in the definition of *RatioOfSumOfCosts*, there is an arrow from *SumOfCosts* to *RatioOfSumOfCosts*. *SumOfCosts* is linked to cubes *DrugPrescription*, *AmbTreatment*, and *Hospitalization*. Thus, also *RatioOfSumOfCosts* can be applied to these cubes. The second score *RatioOfAvgCostsPerInsurant* is defined by arithmetic expression $\text{CoI} . \text{AvgCostsPerInsurant} / \text{CoC} . \text{AvgCostsPerInsurant}$ which computes the ratio of the average costs per insurant of the context of interest and the average costs per insurant of the context of comparison. Also this score can be applied to cubes *DrugPrescription*, *AmbTreatment*, and *Hospitalization*.

2.4.2.6 Score Predicates

Similarly to aggregate measure predicates, score predicates are defined. Score predicates filter the result set of comparative queries.

Definition 2.19. A *score predicate* is a boolean expression defined over a set of scores and possibly aggregate measures qualified by *CoI* and *CoC*. \square

As for aggregate measure predicates, we assume to have SQL based notations for boolean expressions that define score predicates such that these expressions can be considered as a part of an SQL-having-clause. Figure 2.3

presents a score predicate defined by the expression *RatioOfAvgCostsPerInsurant* > 1. We also allow to use additionally aggregate measures within the defining boolean expression. Score predicates also obtain names.²⁸ In the example of Figure 2.3, the score predicate is named as *IncreasedAvgCostsPerInsurant*. Graphically, score predicates are also depicted as rounded rectangles containing the name in the upper part and the defining expression in the lower part. Similarly to the visualization of aggregate measure predicates, the name of a score predicate is prefixed by a filter symbol as a pictogram which associates the filtering of a query result set. In contrast to aggregate measure predicates, this filter pictogram includes a small score symbol.

Again, implications are used to define subsumption hierarchies (not depicted in Figure 2.3) that can be used by analysts to narrow or broaden restrictions on query result sets.

Definition 2.20. If F_1 and F_2 are score predicates and F_1 implies F_2 , we write $F_1 \Rightarrow F_2$ and also say F_2 *subsumes* F_1 . In this case, F_2 is the *subsumer* of F_1 and, reversely, F_1 is the *subsumee* of F_2 . \square

Subsumption relations for score predicates are drawn as arrows with a non-filled arrowhead. Score predicates are linked to scores that are used for the definition. For example, in Figure 2.3, the definition of score predicate *IncreasedAvgCostsPerInsurant* encloses score *RatioOfAvgCostsPerInsurant*. Hence, a line from score predicate *IncreasedAvgCostsPerInsurant* to score *RatioOfAvgCostsPerInsurant* is drawn. If there is a subsumption arrow from one to another score predicate, a line from the source predicate to the score used for defining the score predicate is omitted. A score predicate can be applied to cubes, if all scores and aggregate measures of the definition of the predicate are parts of that cube. As depicted in Figure 2.3, score predicate *IncreasedAvgCostsPerInsurant* can be applied to cubes *DrugPrescription*, *AmbTreatment*, and *Hospitalization*.

At instance level, score predicates are 1-ary predicates defined over the

²⁸Again, the name and the recursively resolved expression of a score predicate can be obtained by *NameOf* and *ExprOf*.

records of a result set of a comparative query. For a record of such a result set, the score predicate is true, if the defining expression of the score predicate evaluates to true for this record. If, for a record of the comparative query result set, the value of score *RatioOfAvgCostsPerInsurant* is greater than 1 (i.e., there is an increase of average costs per insurant), the record is filtered, else it will be omitted.

2.4.3 Cube Schemas and Instances

In this section, we introduce the concepts of cube schemas and cube instances. These concepts are based on fact schemas and fact instances of dimensional fact models as presented in [34]. Moreover, we add enrichments as introduced in the previous subsections.

2.4.3.1 Cube Schemas

A cube schema corresponds to the notion of a fact schema in [34]. It comprises dimension schemas and base measures. For APMN4BI, additional constructs are introduced as an enrichment of a DFM: dimensional operators and dimensional predicates as enrichments for dimension schemas, derived base measures, base measure predicates, simple and derived aggregate measures, aggregate measure predicates, scores, and score predicates.

Definition 2.21. A *cube schema* $C = (DimSchemas_C, BMsrs_C, BMsrPredicates_C, AMsrs_C, AMsrPredicates_C, Scores_C, ScorePredicates_C)$ comprises

1. a finite non-empty set of dimension schemas $DimSchemas_C$, such that if $D, D' \in DimSchemas_C$ and $D \neq D'$, then $Lvls_D \cap Lvls_{D'} = \emptyset$ and $DescrAttrs_D \cap DescrAttrs_{D'} = \emptyset$,
2. a closed finite set of base measures $BMsrs_C$ such that all numeric descriptive attributes of $BMsrs_C$ belong to a dimension schema in $DimSchemas_C$,
3. a finite set of base measure predicates $BMsrPredicates_C$ defined over $BMsrs_C$,

4. a closed finite set of aggregate measures $AMsrs_C$ such that all base measures and all dimension levels used for defining simple aggregate measures in $AMsrs_C$ are elements of $BMsrs_C$ or belong to a dimension schema in $DimSchemas_C$, respectively,
5. a finite set of aggregate measure predicates $AMsrPredicates_C$ defined over $AMsrs_C$,
6. a finite set of scores $Scores_C$ such that all aggregate measures used for defining scores in $Scores_C$ are elements of $AMsrs_C$, and
7. a finite set of score predicates $ScorePredicates_C$ defined over $Scores_C$.

Furthermore, $SimpleBMsrs_C$ contains all simple base measures of $BMsrs_C$, $DerivedBMsrs_C$ contains all derived base measures of $BMsrs_C$, $DescrAttrBMsrs_C$ contains all descriptive attributes of $BMsrs_C$, $SimpleAMsrs_C$ contains all simple aggregate measures of $AMsrs_C$, and $DerivedAMsrs_C$ contains all derived aggregate measures of $AMsrs_C$. \square

A cube schema obtains a name (by function $NameOf$) that can be used to refer to the cube schema. In the graphical notation (see Figure 2.3), the component of a cube schema that contains simple base measures is depicted as a red-colored rectangle decorated with a “cube symbol”. It is separated into two parts, the upper one containing the name of a cube schema and the lower one comprising the simple base measures of the cube schema. Such rectangles are linked by a line to the graphical notation of dimension schemas that belong to the cube schema. If simple base measures listed in this rectangle are used in the definition of a derived base measure or in the definition of a simple aggregate measure, then there is also a line between the “cube schema rectangle” and the rectangle representing the derived base measure or the simple aggregate measure.

In our running example (see Figure 2.3), we have cube schemas $DrugPrescription$, $AmbTreatment$, and $Hospitalization$ for the business events (facts) “drug prescription”, “ambulant treatment”, and “hospitalization”. $DrugPrescription$ and $AmbTreatment$ comprise simple base measures $quantity$

and *costs*. Cube schema *Hospitalization* contains simple base measures *days* and *costs*. Dimension schemas *Time* and *Insurant* are part of all three cube schemas and thus are connected to them by a line, dimension schema *Doctor* is part of cube schemas *DrugPrescription* and *AmbTreatment*. *Drug* is a dimension schema of *DrugPrescription*, dimension schema *MedService* of *AmbTreatment*, and dimension schema *Hospital* of *Hospitalization*.

To simplify the definitions of APMN4BI and to simplify the query generation to SQL, we do not allow dimension roles in a cube schema. Dimension roles must be implemented as separate dimensions which does not restrict the APMN4BI approach. If one wants to distinguish, for instance, between dimension roles “prescribing date” and “billing date” in cube schema *DrugPrescription*, we would define two different time dimension schemas instead of using the dimension schema *Time* twice.

In Figure 2.3, derived base measure *costsPerUnit* is defined by simple base measures *costs* and *quantity*, and thus there is a line from it to the rectangles of both cube schemas *DrugPrescription* and *AmbTreatment*. Similarly, simple aggregate measures *SumOfQuantity* and *SumOfCosts* use simple base measures *quantity* and *costs* of cube schemas *DrugPrescription* and *AmbTreatment* which again is symbolized by connecting lines. Simple base measures *days* and *costs* of cube schema *Hospitalization* are parts of the definition of simple aggregate measures *SumOfDays* and *SumOfCosts*, respectively, again indicated by connecting lines.

Note, that a cube schema comprises further constituents like base measure predicates, derived aggregate measures (beside simple aggregate measures), aggregate measure predicates, scores, score predicates, or also components of connected dimension schemas as, for instance, dimensional operators and dimensional predicates. All these concepts have been introduced in the previous sections.

2.4.3.2 Cube Instances

A cube instance of a cube schema represents business events denoted as facts²⁹ such that one fact is associated with dimension nodes and values of simple base measures. The cube schema determines the dimension schemas of the dimension instances of the associated dimension nodes and the simple base measures of the associated values.

Definition 2.22. Let $C = (DimSchemas_C, BMsrs_C, BMsrPredicates_C, AMsrs_C, AMsrPredicates_C, Scores_C, ScorePredicates_C)$ be a cube schema with $DimSchemas_C = \{D_1, \dots, D_n\}$ and $SimpleBMsrs_C = \{b_1, \dots, b_m\}$.

A cube instance $\mathbf{c} = (C, DimInstances_{\mathbf{c}}, Facts_{\mathbf{c}})$ of cube schema C comprises

1. a set of dimension instances defined as $DimInstances_{\mathbf{c}} = \{\mathbf{d}_1, \dots, \mathbf{d}_n\}$ such that, for all $1 \leq i \leq n$, \mathbf{d}_i is a dimension instance of D_i , and
2. a finite non-empty set $Facts_{\mathbf{c}}$ (the set of facts of \mathbf{c}) such that $Facts_{\mathbf{c}} \subseteq BaseNodes_{\mathbf{d}_1} \times \dots \times BaseNodes_{\mathbf{d}_n} \times dom(b_1) \times \dots \times dom(b_m)$ where, for $1 \leq i \leq m$, $dom(b_i)$ denotes the value domain of simple base measure b_i .

Moreover, we define the following: $CubeSchema_{\mathbf{c}} = C$; $DimSchemas_{\mathbf{c}} = DimSchemas_C$; for $1 \leq i \leq n$, $DimInstance_{\mathbf{c}}(D_i) = \mathbf{d}_i$ and $Nodes_{\mathbf{c}}(D_i) = Nodes_{\mathbf{d}_i}$; $BMsrs_{\mathbf{c}} = BMsrs_C$; $BMsrPredicates_{\mathbf{c}} = BMsrPredicates_C$; $AMsrs_{\mathbf{c}} = AMsrs_C$; $AMsrPredicates_{\mathbf{c}} = AMsrPredicates_C$; $Scores_{\mathbf{c}} = Scores_C$; $ScorePredicates_{\mathbf{c}} = ScorePredicates_C$. \square

Analogously to dimension instances, note that a cube schema is a part of the definition of a cube instance. If there are two cube instances \mathbf{c}_1 and \mathbf{c}_2 of cube schema C_1 and C_2 , respectively, and $C_1 \neq C_2$, then also $\mathbf{c}_1 \neq \mathbf{c}_2$, even if $DimInstances_{\mathbf{c}_1} = DimInstances_{\mathbf{c}_2}$ and $Facts_{\mathbf{c}_1} = Facts_{\mathbf{c}_2}$. This would be the case, if cube schemas C_1 and C_2 only differ in derived base measures, base measure predicates, aggregate measures, aggregate measure predicates,

²⁹In [34], also referred to as *primary fact instances*.

scores, or score predicates—all of them are components that only exists at schema level and that can be applied to cube instances.

Figure 2.4 and Figure 2.5 show an excerpt of some facts of cube instance `DrugPrescription` of cube schema *DrugPrescription* of Figure 2.3.³⁰ Cube instances are also named by function *NameOf*. For simplicity, we use the same name for both cube schema and cube instance and also for dimension schemas and dimension instances (but in both cases with different character fonts). In Figure 2.4, the cube instance including dimension instances is depicted in an abstract manner, whereas Figure 2.5 represents the cube instance in a relational manner as dimension tables (blue color) and one fact table (red color). The example shows two dimension instances `Time` and `Doctor` (elements of $DimInstances_{DrugPrescription}$) of dimension schemas *Time* and *Doctor* (elements of $DimSchemas_{DrugPrescription}$). For both dimension instances, some dimension nodes are depicted, for example, dimension nodes `1 May 2015` at dimension level *date*, `May 2015` at dimension level *month*, `Dr. No` at dimension level *doctor*, or `Linz-Stadt` at dimension level *docDistrict*. Lines between two dimension nodes represent a sub-super-node-relationship. For instance `Dr. No` is a subnode of `Linz-Stadt`. A fact of cube instance `DrugPrescription` comprises values of simple base measures *quantity* and *costs*. The representation of facts are considered as 6-tuples comprising four base nodes (one for each dimension instance) and two simple base measure values (for *quantity* and *costs*). Figure 2.5 presents four of such 6-tuples: (1 May 2015, Dr. No, ..., ..., 1, 100), (1 May 2015, Dr. Marbuse, ..., ..., 2, 50), (2 May 2015, Dr. No, ..., ..., 1, 10), (2 May 2015, Dr. Marbuse, ..., ..., 3, 5). Note that base nodes of dimension instances `Drug` and `Insurant` are omitted in the visualization of this example. In the abstract visualization of Figure 2.4, these 6-tuples can be obtained by combining the depicted base nodes and the graphically associated base measures. The following subsection introduces the definition of a star schema that forms the basis of a relational implementation of cubes.

Dimensional operators, dimensional predicates, derived base measures, base measure predicates, aggregate measures, aggregate measure predicates,

³⁰As for dimension instance, names for cube instances are written in a sans serif font to distinguish from names for cube schemas which are written in a *slant* font.

scores, and score predicates are considered as expressions at schema level (as introduced in previous sections) that are evaluated at instance level. In this thesis, we assume to have SQL-like expressions at schema level that are evaluated in an SQL-like manner at instance level. Complete SQL statements can be constructed by mapping analysis situations into SQL (see subsequent chapters) where the mentioned expressions above are used. In this sense, semantics of analysis situations is defined via SQL.

2.4.4 Translation into a Star Schema

In this subsection, we introduce a possible translation of a cube schema or a cube instance into a relational representation in the form of a star schema. This translation provides a basis to define the semantics of analysis situations which is founded on translation into SQL queries.

A cube schema induces relational schemas which represent the tables of a star schema that defines a dimension table for each dimension schema and a fact table containing base measures and the relationship to dimension nodes. In this thesis, base nodes are used as primary keys in dimension tables and as foreign keys in fact tables.³¹ The following two definitions formalize the star schema induced by a cube schema or induced by a cube schema of a cube instance.

Definition 2.23. Given a cube schema C , we define a relational schema $FactTbl_C$ (*fact table*) identified by name $NameOf(C)$ and, for each $D \in DimSchemas_C$, a relational schema $DimTbl_D$ (*dimension table*) identified by name $NameOf(D)$, such that,

1. for each $L \in Lvls_D - \{top_D\}$, $NameOf(L)$ is an attribute of $DimTbl_D$,
2. for each $A \in DescrAttrs_D$, $NameOf(A)$ is an attribute of $DimTbl_D$,
3. $NameOf(base_D)$ is an attribute of $FactTbl_C$ used for foreign keys that references primary keys in $DimTbl_D$, and,

³¹Instead of surrogate keys, we use base nodes as primary and foreign keys to simplify further formalizations.

4. for each $b \in BMsrs_C$, $NameOf(b)$ is an attribute of $FactTbl_C$. \square

Whereas the previous definition specifies dimension tables and fact tables on the basis of a cube schema, the subsequent one formalizes the same on the basis of cube instances. Finally, this second definition also uses the information of the underlying cube schema of the cube instance. The only difference between both definition is that the table names are derived either from cube schemas or dimension schemas (in the previous definition) or from cube instances or dimension instances (in the following definition).

Definition 2.24. Given a cube instance \mathbf{c} , we define a relational schema $FactTbl_{\mathbf{c}}$ (*fact table*) identified by name $NameOf(\mathbf{c})$ and, for each $\mathbf{d} \in DimInstances_{\mathbf{c}}$, a relational schema $DimTbl_{\mathbf{d}}$ (*dimension table*) identified by name $NameOf(\mathbf{c})$, such that,

1. for each $L \in Lvl_{\mathbf{d}} - \{top_{\mathbf{d}}\}$, $NameOf(L)$ is an attribute of $DimTbl_{\mathbf{d}}$,
2. for each $A \in DescrAttrs_{\mathbf{d}}$, $NameOf(A)$ is an attribute of $DimTbl_{\mathbf{d}}$,
3. $NameOf(base_{\mathbf{d}})$ is an attribute of $FactTbl_{\mathbf{c}}$ used for foreign keys that references primary keys in $DimTbl_{\mathbf{d}}$, and,
4. for each $b \in BMsrs_{\mathbf{c}}$, $NameOf(b)$ is an attribute of $FactTbl_{\mathbf{c}}$. \square

In both definitions, a dimension table comprises all dimension levels as attributes except the top level which can only have dimension node **all** as value at instance level. Additionally, all descriptive attributes also become attributes of a dimension table. Finally, a fact table is defined by base levels and base measures. It depends on the situation which definition can be applied. If one only has a cube schema, the first definition has to be used. In the other case, if one has a cube instance, also the second definition can be applied.

Notice that both definitions represent a specific translation of a cube schema into a star schema that will be used later to define the semantics of analysis situations. For simplicity, this translation omits attributes that can be found in other kinds of transformations. This translation does not

consider, for instance, surrogate keys for dimension records or additional attributes implementing historization.

The excerpt of the example in Figure 2.5 is presented in accordance with the above definitions. Cube schema *DrugPrescription* induces a relational schema $FactTbl_{DrugPrescription}$ identified by name $NameOf(DrugPrescription)$ which again, in our example, is written at schema level as *DrugPrescription* and at instance level by another font style as **DrugPrescription**. Similarly, one can see relational schemas $DimTbl_{Time}$ and $DimTbl_{Doctor}$ induced from dimension schemas *Time* and *Doctor*. These dimension tables obtain names $NameOf(Time) = Time$ and $NameOf(Doctor) = Doctor$, at instance level written by another font style as **Time** and **Doctor**. Attributes *date*, *month*, *quarter*, and *year* of dimension table **Time** are induced from the dimension levels of dimension schema *Time*, and attributes *doctor*, *docAge*, *docDistrict*, *inhPerSqkmInDocDistr*, and *docProvince* of dimension table **Doctor** are induced from the dimension levels and descriptive attributes of dimension schema *Doctor*. Note, there is no attribute for dimension level *top* in both cases. Fact table **DrugPrescription** has base levels *date*, *doctor*, *insurant*, and *drug*, and base measures *quantity* and *costs* as attributes.

In the subsequent chapter, this relational representation as a star schema is used to define the semantics of analysis situations. Similarly to eDFM, analysis situations are introduced conceptually and, in a second step, the semantics is defined on the basis of SQL.

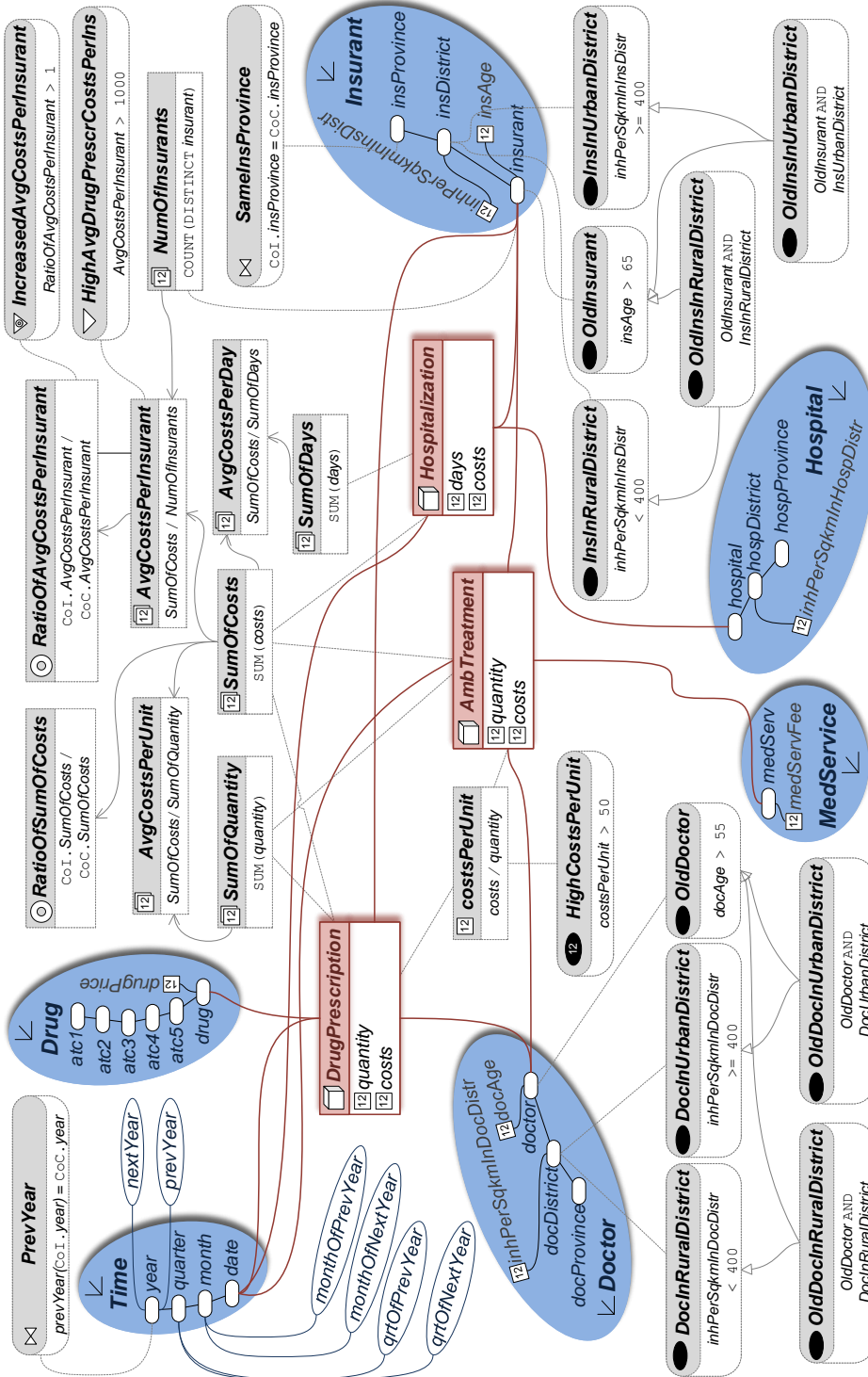


Figure 2.3: eDFM of the running example

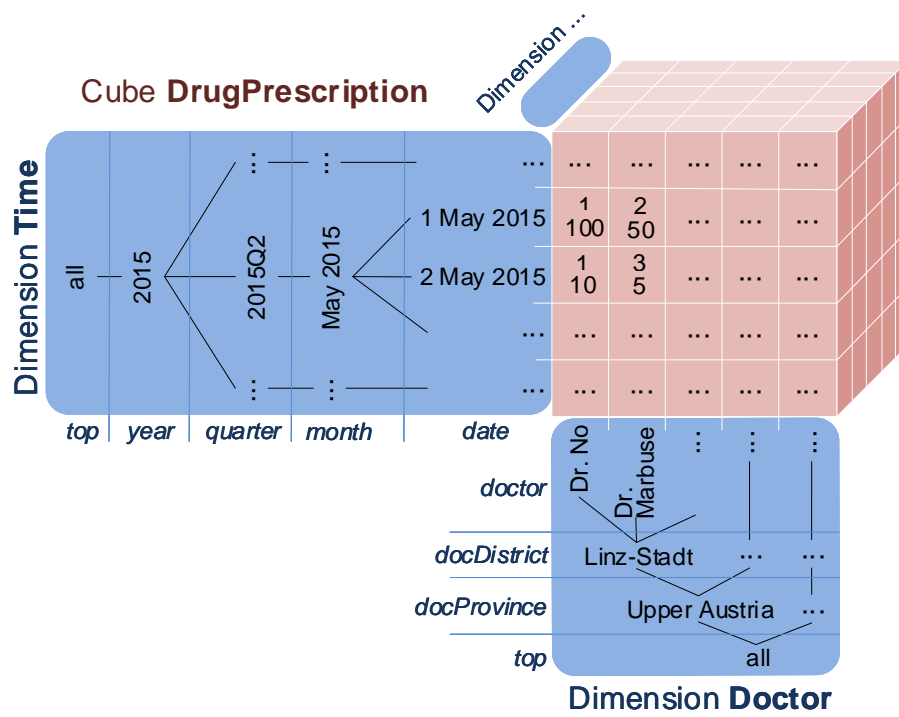


Figure 2.4: Abstract visualization of an example of a cube instance

<i>date</i>	<i>month</i>	<i>quarter</i>	<i>year</i>
...
1 May 2015	May 2015	2015Q2	2015
2 May 2015	May 2015	2015Q2	2015
...

<i>doctor</i>	<i>docAge</i>	<i>docDistrict</i>	<i>inhPerSqkmInDocDistr</i>	<i>docProvince</i>
...
Dr. No	56	Linz-Stadt	2115	Upper Austria
Dr. Marbuse	49	Linz-Stadt	2115	Upper Austria
...

<i>date</i>	<i>doctor</i>	<i>insurant</i>	<i>drug</i>	<i>quantity</i>	<i>costs</i>
...
1 May 2015	Dr. No	1	100
1 May 2015	Dr. Marbuse	2	50
2 May 2015	Dr. No	1	10
2 May 2015	Dr. Marbuse	3	5
...

Figure 2.5: Example of a translation of a cube instance into a star schema

Chapter 3

Analysis Situations

Contents

3.1	Non-comparative Analysis Situations	112
3.1.1	Dimension Qualification	112
3.1.2	Formal Definition	114
3.1.3	Graphical Representation	117
3.1.4	Translation into SQL	123
3.1.5	Discussion	126
3.2	Comparative Analysis Situations	127
3.2.1	Preliminary Remarks	127
3.2.2	Formal Definition	129
3.2.3	Graphical Representation	133
3.2.4	Translation into SQL	139
3.2.5	Discussion	144

We conceptually introduce multi-dimensional queries and comparisons performed by business analysts. Non-comparative analysis situations are used to model multi-dimensional queries. Comparison of query results of two non-comparative analysis situations can be specified by comparative analysis situations. The notion “analysis situation” comprises both non-comparative and comparative analysis situation. We present formal definitions, graphical

representations, and translations into SQL queries that define semantics of analysis situations.

3.1 Non-comparative Analysis Situations

Non-comparative analysis situations represent multi-dimensional queries on cube instances. A user specifies measures to be calculated, restricts queries with respect to base measure values, selects dimension nodes for which measures have to be computed and aggregated, specifies the granularity at which measures have to be aggregated, and, finally, she or he filters interesting rows of the query result set.

In the first subsection, we present dimension qualifications as a prerequisite. Subsequently, formal definition, graphical representation, and translation into SQL queries are given.

3.1.1 Dimension Qualification

A dimension qualification refers to a single dimension. It describes the selection of dimension nodes that have to be analyzed. This is done by giving a dice node (and a dice level) and optional slice conditions (expressed by dimensional predicates). The dice node is a dimension node such that it and all its subnodes are selected for analysis, if they additionally fulfill all slice conditions. Finally, the granularity level specifies at which aggregation level measure results have to be listed.

Definition 3.1. A *dimension qualification* $DQ = (DimSchema_{DQ}, DiceLvl_{DQ}, DiceNode_{DQ}, SliceConds_{DQ}, GranLvl_{DQ})$ comprises

1. a dimension schema $DimSchema_{DQ}$ (subsequently abbreviated as D),
2. a *dice level* $DiceLvl_{DQ} \in Lvl_s_D$,
3. a *dice node* $DiceNode_{DQ} \in Nodes_{\mathbf{d}}(DiceLvl_{DQ})$ for a $\mathbf{d} \in DimInstances_D$,
4. a possibly empty set of *slice conditions* $SliceConds_{DQ} \subseteq DimPredicates_D$,
and

5. a granularity level $GranLvl_{DQ} \in Lvl_s_D$.

Furthermore, for a dimension instance $\mathbf{d} \in DimInstances_D$, the application of DQ to \mathbf{d} is defined as $DQ(\mathbf{d}) = \{\mathbf{n} \in Nodes_{\mathbf{d}} \mid (\mathbf{n} = DiceNode_{DQ} \text{ or } \mathbf{n} \rightarrow DiceNode_{DQ})\}$ and, for all $P \in SliceConds_{DQ}$, P is defined at node \mathbf{n} and P is true for \mathbf{n} . \square

A dimension qualification DQ specifies a schema for selecting dimension nodes of an instance of dimension schema $DimSchema_{DQ}$. Dice node $DiceNode_{DQ}$ and all its subnodes are selected, if the nodes satisfy all dimension predicates in the set of slice conditions $SliceConds_{DQ}$. In the case of dice node **all** at dice level *top*, all nodes of the dimension instance satisfying the slice conditions are chosen. The set $SliceConds_{DQ}$ can be considered as an overall slice condition consisting of a conjunction of all dimensional predicates of $SliceConds_{DQ}$. If $SliceConds_{DQ}$ is empty, the overall slice condition represents truth value *true*. Granularity level $GranLvl_{DQ}$ specifies the level a measure has to be aggregated. In the case of $GranLvl_{DQ} = top$, only one single aggregation value can be obtained.

The dimension qualification DQ can be applied to a dimension instance \mathbf{d} of dimension schema $DimSchema_{DQ}$. Set $DQ(\mathbf{d})$ represents all selected dimension nodes obtained by applying dimension qualification DQ . Note, that granularity level $GranLvl_{DQ}$ is not yet used in the definition of $DQ(\mathbf{d})$. It is needed just in analysis situations to denote the granularity level (with respect to dimension instance \mathbf{d}) for measure aggregation.

Three examples of dimension qualifications based on the eDFM in Figure 2.3 are presented in this and the subsequent paragraphs. Dimension qualification $InsDQ1 = (Insurant, insProvince, \text{Upper Austria}, \{InsInRuralDistrict\}, insDistrict)$ comprises dimension schema $DimSchema_{InsDQ1} = Insurant$, dice level $DiceLvl_{InsDQ1} = insProvince$, dice node $DiceNode_{InsDQ1} = \text{Upper Austria}$, a set $SliceConds_{InsDQ1} = \{InsInRuralDistrict\}$ containing the single slice condition $InsInRuralDistrict$, and granularity level $GranLvl_{InsDQ1} = insDistrict$. Dimensional predicate $InsInRuralDistrict$ is defined over nodes of levels $insDistrict$ and $insurant$. Applied to an instance \mathbf{d} of dimension schema $Insurant$ with $\text{Upper Austria} \in Nodes_{\mathbf{d}}$, set $InsDQ1(\mathbf{d})$ contains all nodes of

levels *insDistrict* and *insurant* that are subnodes of **Upper Austria** provided that dimension predicate *InsInRuralDistrict* is true for these subnodes. This dimension qualification restricts an analysis to rural insurants' districts of Upper Austria. Additionally, granularity level $GranLvl_{InsDQ1}$ specifies that measures have to be aggregated at the level of insurants' districts.

The second example of a dimension qualification is defined as $InsDQ2 = (Insurant, insProvince, \text{Upper Austria}, \{InsInRuralDistrict, OldInsurant\}, insDistrict)$. As a difference to the first example, the set of slice conditions $SliceConds_{InsDQ2}$ comprises two dimensional predicates: *InsInRuralDistrict* and *OldInsurant*. Because slice condition *OldInsurant* is only defined on nodes of level *insurant*, set $InsDQ2(\mathbf{d})$ only contains nodes of level *insurant* (for some instance \mathbf{d} of dimension schema *Insurant* with $\text{Upper Austria} \in Nodes_{\mathbf{d}}$). This means that only old insurants of Upper Austria living in rural districts are selected for analysis. Alternatively to this set of slice conditions, one can replace it by a set only containing the single dimensional predicate *OldInsInRuralDistrict* which is defined as *OldInsurant* AND *InsInRuralDistrict*.

In the last example, we define dimension qualification $InsDQ3 = (Insurant, top, all, \emptyset, top)$ that can be used to select all nodes of a dimension instance of dimension schema *Insurant*. Dice node *all* at dice level *top* comprises all dimension nodes of an instance of dimension schema *Insurant* and because of $SliceConds_{insDQ3} = \emptyset$, there are no further restrictions. Dimension level *top* is also used for specifying granularity level $GranLvl_{InsDQ3}$ meaning that measures are aggregated over all nodes yielding a single measure value.

3.1.2 Formal Definition

A non-comparative analysis situation refers to a cube instance that describes the data to be queried. The dimension nodes of the dimension instances that are selected in the query are specified by dimension qualifications. Facts can be additionally restricted by base measure predicates. Furthermore, a non-comparative analysis situation has to specify the aggregate measures to be calculated. Optionally, filter conditions can be applied to restrict a query

result.¹

Definition 3.2. A non-comparative analysis situation $\mathbf{as} = (\mathbf{c}, B, M, DQS, F)$ comprises

1. a cube instance \mathbf{c} of cube schema $CubeSchema_{\mathbf{c}}$,
2. a possibly empty set of base measure conditions $B \subseteq BMrPredicates_{\mathbf{c}}$,
3. a non-empty set of aggregate measures $M \subset AMsr_{\mathbf{c}}$,
4. a set of dimension qualifications DQS such that for each $D \in DimSchemas_{\mathbf{c}}$ there exists a unique $DQ \in DQS$ with $DimSchema_{DQ} = D$, and
5. a possibly empty set of filter conditions $F \subseteq AMsrPredicates_{\mathbf{c}}$.

Moreover, in the context of \mathbf{as} , we define the following: $CubeInstance_{\mathbf{as}} = \mathbf{c}$, $CubeSchema_{\mathbf{as}} = CubeSchema_{\mathbf{c}}$, $BMrConds_{\mathbf{as}} = B$, $AMsr_{\mathbf{as}} = M$, $FilterConds_{\mathbf{as}} = F$, $DimQuals_{\mathbf{as}} = DQS$, and $DimSchemas_{\mathbf{as}} = \{DimSchema_{DQ} \mid DQ \in DQS\}$. Furthermore, for $DQ \in DQS$ with $D = DimSchema_{DQ}$, we define: $DimInstance_{\mathbf{as}}(D) = DimInstance_{\mathbf{c}}(D)$, $DiceLvl_{\mathbf{as}}(D) = DiceLvl_{DQ}$, $DiceNode_{\mathbf{as}}(D) = DiceNode_{DQ}$, $SliceConds_{\mathbf{as}}(D) = SliceConds_{DQ}$, and $GranLvl_{\mathbf{as}}(D) = GranLvl_{DQ}$. For $D \in DimSchemas_{\mathbf{c}}$, $DimQual_{\mathbf{as}}(D)$ is the unique $DQ \in DimQuals_{\mathbf{as}}$ with $DimSchema_{DQ} = D$. \square

A non-comparative analysis situation \mathbf{as} selects facts from cube instance $CubeInstance_{\mathbf{as}}$ that are restricted accordingly to base measure conditions $BMrConds_{\mathbf{as}}$. The set $BMrConds_{\mathbf{as}}$ can be considered as an overall base measure condition consisting of a conjunction of all base measure predicates of $BMrConds_{\mathbf{as}}$. In the case of $BMrConds_{\mathbf{as}} = \emptyset$, this overall condition can be considered as true. For the resulting subset of facts, aggregate measures of set $AMsr_{\mathbf{as}}$ are computed and aggregated for dimension nodes which are specified by the dimension qualifications of set $DimQuals_{\mathbf{as}}$. The dimension

¹In this chapter, we introduce analysis situations as query instances in an analysis process that are applied to cube instances. Thus, similarly to dimension and cube instances, we use a **bold** font (for example, \mathbf{as}) to denote analysis situations.

qualifications also define the granularity levels at which measures are aggregated. Note, that for each dimension schema of $CubeInstance_{as}$, there exists a unique dimension qualification. Finally, the result set can be filtered optionally by filter conditions of set $FilterConds_{as}$ that examine calculated measure values and only those rows are selected that satisfy all filter conditions. As for base measure and slice conditions, one can consider set $FilterConds_{as}$ as an overall filter condition which is specified as a conjunction of all measure predicates in $FilterConds_{as}$. If $FilterConds_{as}$ is empty, the overall filter condition represents truth value *true*.

The formal semantics of an analysis situation is defined by a mapping into SQL which will be presented later in this chapter. In this subsection we give further explanations on the basis of an example. We assume a business analyst wants to select high drug prescription costs in rural insurants' districts in year 2016 listed per insurance province having high drug prescription costs per insurant. This vague verbalization of a query can be expressed conceptually in a precise way using the notion of a non-comparative analysis situation $as = HighDrugPrescrCostsInRuralDistrPerProv$:²

- $CubeInstance_{as} = DrugPrescription$
- $BMSrConds_{as} = \{HighCostsPerUnit\}$
- $AMsr_{as} = \{SumOfCosts, AvgCostsPerInsurant\}$
- $DimQuals_{as} = \{(Time, year, 2016, \emptyset, top), (Insurant, top, all, \{InsInRuralDistrict\}, insProvince), (Drug, top, all, \emptyset, top), (Doctor, top, all, \emptyset, top),\}$
- $FilterConds_{as} = \{HighAvgDrugPrescrCostsPerIns\}$

In this example, facts from cube instance **DrugPrescription** are selected, if their measure values fulfill base measure predicate *HighCostsPerUnit*, i.e., facts that satisfy expression *costs / quantity > 50* are selected. Only dimension nodes specified by dimension qualifications are selected for calculation

²Similarly to dimension instances and cube instances, we use a sans serif font for naming analysis situations.

of aggregate measures. In dimension *Time*, dice node *2016* of dice level *year* and all subnodes (for example, nodes *2016Q1* and *2016Q2* of level *quarter*, nodes *January 2016* and *February 2016* of level *month*, or nodes *1 January 2016* and *2 January 2016* of level *date*) are selected for aggregation. There are no further restrictions stated as slice conditions. In dimension *Insurant*, all dimension nodes are considered by dice node *all* at dice level *top* but further restriction is applied by slice condition *InsInRuralDistrict*, i.e., only those nodes are selected for measure computation that satisfy expression *inhPerSqkmInInsDistr* < 400. Dimension qualifications for dimensions *Drug* and *Doctor* select all dimension nodes because of dice node *all* at dice level *top* and because there is no slice condition for both. The query result set of the analysis situation comprises rows with aggregate measures *SumOfCosts* and *AvgCostsPerInsurant* (as columns) which are aggregated at levels that are specified by the granularity levels also contained in the dimension qualifications. In dimension *Insurant*, we have granularity level *insProvince*. All other dimension qualifications have granularity level *top*. Thus a result row consists of a dimension node referring an insurants' province for dimension *Insurant*, node *all* for other dimensions, and numeric values for aggregate measures *SumOfCosts* and *AvgCostsPerInsurant*. Additionally, this result set is restricted by filter condition *HighAvgDrugPrescrCostsPerIns*, i.e., only those result rows that satisfy condition *AvgCostsPerInsurant* > 1000 are transferred to the user.

3.1.3 Graphical Representation

Although the main contribution of this work lies in the constructs of APMN-4BI themselves and not in the visual design of them, we give graphical representations of non-comparative analysis situations. One representation (Figure 3.1 and Figure 3.2) has an obvious one-to-one correspondence to the formal definition (*full representation*). A second representation (Figure 3.3 and Figure 3.4) makes some visual simplifications and sets a focus on the bare essentials (*lean representation*). The third one (Figure 3.5 and Figure 3.6) is a condensed form that hides details (*condensed representation*)—it can be

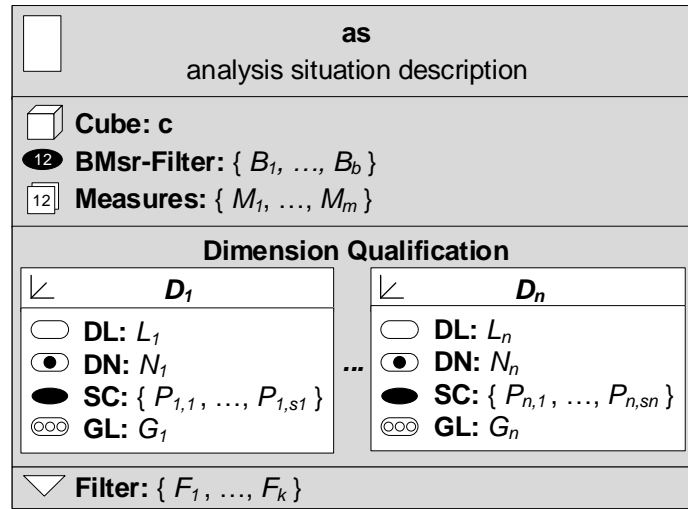


Figure 3.1: Template of a non-comparative analysis situation depicted in full graphical representation

used only in combination with navigation operators which will be explained later.

Figure 3.1 shows a formal template of the graphical representation of a non-comparative analysis situation. In the figure, **as** denotes a name for the analysis situation which is positioned on the top of the graphical representation and which is followed by an optional description. The rectangle in the top left corner is used as a pictogram for a non-comparative analysis situation. For each constituent of the analysis situation, we draw another pictogram accompanied by a short label:³ label **Cube** for the cube instance, label **BMSr-Filter** for base measure conditions, label **Measures** for aggregate measures, and label **Filter** for filter conditions of the non-comparative analysis situation. The name of the dimension schema of each dimension qualification is prefixed by a specific pictogram (without any label) representing dimension axis in a figurative sense. The components of a dimension qualification again obtain a pictogram and a label symbolizing its general

³Pictograms used for analysis situations have been designed equal or at least similar to pictograms used in an eDFM with respect to the corresponding elements that are represented by a respective pictogram.

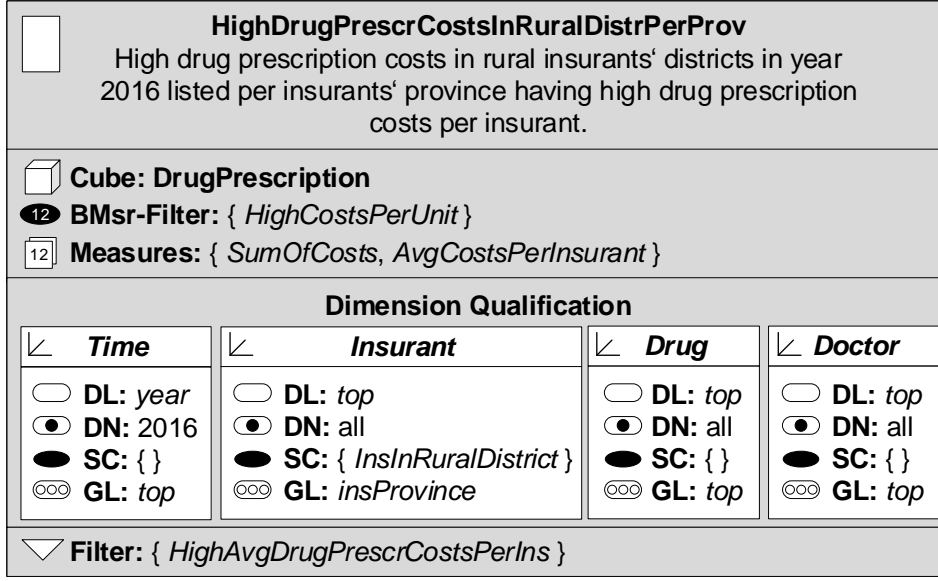


Figure 3.2: Example of a non-comparative analysis situation depicted in full graphical representation

purpose: label **DL** introduces the dice level, label **DN** the dice node, label **SC** the set of slice conditions, and label **GL** the granularity level of a dimension qualification.

All constituents of Definition 3.2 can be found in the graphical representation of Figure 3.1: $CubeInstance_{as} = \mathbf{c}$, $BMsrConds_{as} = \{B_1, \dots, B_b\}$, $AMsrs_{as} = \{M_1, \dots, M_m\}$, $FilterConds_{as} = \{F_1, \dots, F_k\}$, and $DimSchemas_{as} = \{D_1, \dots, D_n\}$. Moreover, for $1 \leq i \leq n$, $DiceLvl_{as}(D_i) = L_i$, $DiceNode_{as}(D_i) = N_i$, $SliceConds_{as}(D_i) = \{P_{i,1}, \dots, P_{i,s_i}\}$, and $GranLvl_{as}(D_i) = G_i$.

Note, for conformity and clarity, also dice level *top*, dice node **all**, and granularity level *top* are stated in the full graphical representation. The base measure conditions, the slice conditions, and the filter conditions result in the overall base measure condition $B_1 \wedge \dots \wedge B_b$, for $1 \leq i \leq n$, in the overall slice condition $P_{i,1} \wedge \dots \wedge P_{i,s_i}$, and in the overall filter condition $F_1 \wedge \dots \wedge F_k$. If there are no base measure conditions, no slice conditions,

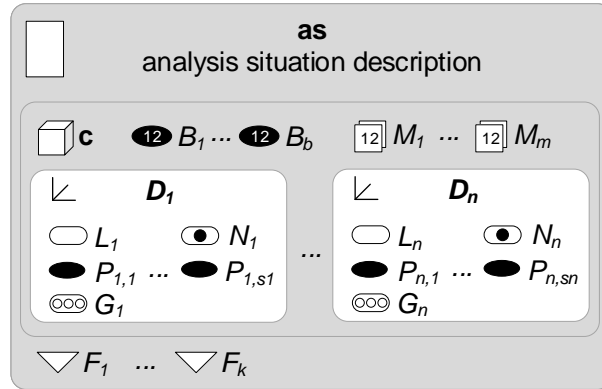


Figure 3.3: Template of a non-comparative analysis situation depicted in lean graphical representation

or no filter conditions, we write an empty set⁴, respectively. In this case the overall base measure condition, the overall slice condition, and the overall filter condition becomes true.

Figure 3.2 demonstrates the full graphical representation of the example of a non-comparative analysis situation presented in Section 3.1.2. Notice, as already mentioned generally, we also draw in (as default values) dice level *top* and dice node *all* in the dimension qualifications for dimension schemas *Insurant*, *Drug*, and *Doctor*. The dimension qualifications for dimension schemas *Time*, *Drug*, and *Doctor* contain as default values granularity level *top* and the empty set for the set of slice conditions.

The lean graphical notation of a non-comparative analysis situation focuses on the substantial information and allows more flexibility in drawing. On the other side, all necessary information is contained such that the lean graphical notation can be transformed in a one-to-one correspondence to a full graphical notation. Figure 3.3 presents a template for a lean graphical notation. The outer frame is drawn as a rounded rectangle. Labels are omitted and only pictograms are used as a graphical prefix beside the constituents. In the case of sets (set of base measure conditions, set of aggregate measures, set of slice conditions, and set of filter conditions), there is

⁴In the empty set notation $\{\}$.

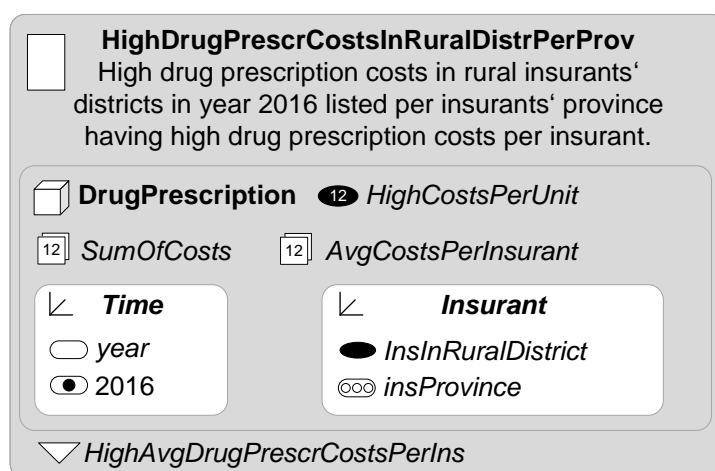


Figure 3.4: Example of a non-comparative analysis situation depicted in lean graphical representation

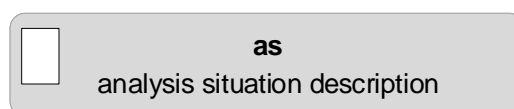


Figure 3.5: Template of a non-comparative analysis situation depicted in condensed graphical representation

no usual set notation but each set element is prefixed by the corresponding pictogram. All constituents can be positioned arbitrary, only membership relation of constituents must be respected via nested rounded rectangles: the outer rounded rectangle containing all constituents; a rounded rectangle containing cube instance, base measure conditions, aggregate measures, and dimension qualifications; a rounded rectangle for each dimension qualification containing dimension schema, dice level, dice node, slice conditions, and granularity level. Default value *top* for dice and granularity levels, default value **all** for dice nodes, and the empty set for base measure conditions, slice conditions, and filter conditions are not stated but they can be considered as implicitly present. Thus each information of the formal definition can be derived from this alternative graphical representation, too.

Figure 3.4 shows exactly the same non-comparative analysis situation as

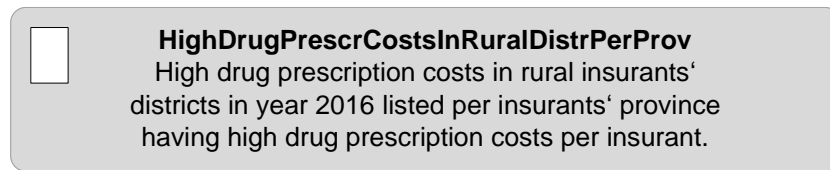


Figure 3.6: Example of a non-comparative analysis situation depicted in condensed graphical representation

depicted in Figure 3.2. All constituents are positioned in a space-saving manner. For instance, the set of aggregate measures is dissolved—both set elements *SumOfCosts* and *AvgCostsPerInsurant* are listed without a specific set notation. In dimension qualification for dimension schema *Time*, the empty set for slice conditions is not depicted explicitly and granularity level *top* (as a default value) can be omitted. Dimension qualification for dimension schema *Insurant* contains default value *top* as dice level and default value *all* as dice node (both not depicted graphically). Because the dimension qualifications for dimension schemas *Drug* and *Doctor* only contain default values (default value *top* for dice levels and granularity levels, default value *all* for dice nodes, and the empty set for slice conditions), both dimension qualifications can be omitted graphically as a whole.

Figure 3.5 shows a template and Figure 3.6 presents an example of a non-comparative analysis situation in a condensed graphical notation. It only comprises the name and an optional description but no constituents of the definition of an analysis situation. Although the example in Figure 3.6 contains the same analysis situation name and the same description like the example in Figure 3.2 (or Figure 3.4), additional information is needed for transforming to the graphical notation of Figure 3.2 (or Figure 3.4). As we will see later, this shape only can be used in combination with navigation operators where information of a target analysis situation can be derived from a source analysis situation and a navigation step.

3.1.4 Translation into SQL

In this subsection, we define the semantics of a non-comparative analysis situation by a mapping into an SQL query. Analysis situations are conceptual constructs for business analysts. The translation into SQL queries represents one option for technical realizations which are based on relational OLAP (ROLAP). The APMN4BI as a conceptual modeling language is not restricted to this type of implementation.

Definition 3.3. Let \mathbf{as} be a non-comparative analysis situation such that $CubeInstance_{\mathbf{as}} = \mathbf{c}$ and $DimInstances_{\mathbf{c}} = \{\mathbf{d}_1, \dots, \mathbf{d}_n\}$, $BMsrsConds_{\mathbf{as}} = \{B_1, \dots, B_b\}$, $AMsrs_{\mathbf{as}} = \{M_1, \dots, M_m\}$, $FilterConds_{\mathbf{as}} = \{F_1, \dots, F_k\}$, $DimSchemas_{\mathbf{as}} = \{D_1, \dots, D_n\}$ with $D_1 = DimSchema_{\mathbf{d}_1}, \dots, D_n = DimSchema_{\mathbf{d}_n}$, $DiceLvl_{\mathbf{as}}(D_1) = L_1, \dots, DiceLvl_{\mathbf{as}}(D_n) = L_n$, $DiceNode_{\mathbf{as}}(D_1) = N_1, \dots, DiceNode_{\mathbf{as}}(D_n) = N_n$, $SliceConds_{\mathbf{as}}(D_1) = \{P_{1,1}, \dots, P_{1,s_1}\}, \dots, SliceConds_{\mathbf{as}}(D_n) = \{P_{n,1}, \dots, P_{n,s_n}\}$, and $GranLvl_{\mathbf{as}}(D_1) = G_1, \dots, GranLvl_{\mathbf{as}}(D_n) = G_n$.

The *semantics of \mathbf{as}* is denoted by $Query_{\mathbf{as}}$ and is defined by an SQL query as presented in the template of Figure 3.7 that is based on the relational schema of cube instance \mathbf{c} .

The result set of an execution of the SQL query that defines the semantics of a non-comparative analysis situation \mathbf{as} is denoted by $ResultSet_{\mathbf{as}}$. The relational schema of result set $ResultSet_{\mathbf{as}}$ is denoted by $ResultSetSchema_{\mathbf{as}}$ and comprises the following attributes: $NameOf(G_1), \dots, NameOf(G_n), NameOf(M_1), \dots, NameOf(M_m)$. \square

Figure 3.7 defines how a non-comparative analysis situation \mathbf{as} can be translated into an SQL query. The given translation assumes that cube and dimension instances are organized in a star schema as relational database tables and that the naming conventions for attributes allow to use natural joins. We use the functions $NameOf$ and $ExprOf$ introduced in the previous chapter at a meta level to return appropriate names and resolved expressions for constructing correct SQL statements. Names for dice and granularity levels and for descriptive attributes are used as names of database table attributes. Aggregate measures, base measure conditions, slice conditions,

```

SELECT  NameOf( $G_1$ ),  $\dots$ , NameOf( $G_n$ ),
        ExprOf( $M_1$ ) AS NameOf( $M_1$ ),  $\dots$ ,
        ExprOf( $M_m$ ) AS NameOf( $M_m$ )
FROM    NameOf(c) NATURAL JOIN
        NameOf(d1) NATURAL JOIN  $\dots$  NATURAL JOIN NameOf(d $n$ )
WHERE   ExprOf( $B_1$ ) AND  $\dots$  AND ExprOf( $B_b$ ) AND
        NameOf( $L_1$ ) = ExprOf( $N_1$ ) AND  $\dots$  AND
        NameOf( $L_n$ ) = ExprOf( $N_n$ ) AND
        ExprOf( $P_{1,1}$ ) AND  $\dots$  AND ExprOf( $P_{1,s_1}$ ) AND  $\dots$  AND
        ExprOf( $P_{n,1}$ ) AND  $\dots$  AND ExprOf( $P_{n,s_n}$ )
GROUP BY NameOf( $G_1$ ),  $\dots$ , NameOf( $G_n$ )
HAVING  ExprOf( $F_1$ ) AND  $\dots$  AND ExprOf( $F_k$ )

```

Additional translation rules: (1) Granularity level *top* is removed from the SELECT and GROUP BY clause. (2) Condition *top* = all is removed from the WHERE clause. (3) An empty WHERE, GROUP BY, or HAVING clause is removed from the SQL statement. (4) If no constituents of a dimension schema are used in the SQL statement, then the corresponding dimension name is removed from the natural join of the FROM clause.

Figure 3.7: Formal specification of the translation of a non-comparative analysis situation into SQL

and filter conditions are resolved with respect to their defining expressions. Furthermore, we assume that these expressions are conform with SQL syntax. Moreover, also dimension nodes have to be stated in correct SQL notation. Names for aggregate measures can be used as alias names in the projection. Finally, as level *top* and node **all** cannot be presented in SQL, corresponding dimension tables and table attributes have to be omitted in the resulting SQL statement which is indicated in Figure 3.7 by additional translation rules.

For illustration only, a template of a query result set of a non-comparative analysis situation **as** is depicted in Figure 3.8. It corresponds to the result set schema *ResultSetSchema*_{as}. The names of the granularity levels G_1, \dots, G_n , and the names of the measures M_1, \dots, M_m are used as column names of the result table.

Figure 3.9 demonstrates the SQL translation (denoted as *Query*_{as}) of analysis situation **as** = HighDrugPrescrCostsInRuralDistrPerProv presented in Figure 3.2. Accordingly to instances of the eDFM of Figure 2.3 (see also

<i>NameOf(G₁)</i>	...	<i>NameOf(G_n)</i>	<i>NameOf(M₁)</i>	...	<i>NameOf(M_m)</i>
...
...
...
...

Figure 3.8: Relational schema of a result set of a non-comparative analysis situation

```

SELECT  insProvince ,
        SUM(costs) AS SumOfCosts ,
        SUM(costs) / COUNT(DISTINCT insurant)
                                                AS AvgCostsPerInsurant

FROM    DrugPrescription NATURAL JOIN
        Time NATURAL JOIN
        Insurant

WHERE   costs / quantity > 50 AND
        year = 2016 AND
        inhPerSqkmInInsDistr < 400

GROUP BY insProvince

HAVING  SUM(costs) / COUNT(DISTINCT insurant) > 1000

```

Figure 3.9: Example of the translation of a non-comparative analysis situation into SQL

Figure 2.4 and Figure 2.5), we assume to have a star schema with fact table *DrugPrescription*, and dimension tables *Time*, *Insurant*, *Drug*, and *Doctor*. To construct the SQL statement of Figure 3.9, table *DrugPrescription* must have attributes *costs*, *quantity*, *date* referring the base level of dimension table *Time*, and *insurant* referring the base level of dimension table *Insurant*. Furthermore, we require to have table columns *date*, *month*, *quarter*, and *year* in dimension table *Time*, and columns *insurant* (serving as a primary key), *insDistrict*, *insProvince*, *insAge*, and *inhPerSqkmInInsDistr* in dimension table *Insurant*. The last two columns represents the descriptive attributes of dimension schema *Insurant*.

A result table of the query of Figure 3.9 comprises columns *insProvince*, *SumOfCosts*, and *AvgCostsPerInsurant* that corresponds to result set schema *ResultSetSchema_{as}* with *as* = *HighDrugPrescrCostsInRuralDistrPerProv*. For

<i>insProvince</i>	<i>SumOfCosts</i>	<i>AvgCostsPerInsurant</i>
Upper Austria	117790612.01	2120.31
Lower Austria	197790612.45	2415.07
...

Figure 3.10: Example of a result set of a non-comparative analysis situation

illustration, a possible result table is depicted in Figure 3.10 containing fictitious data with respect to result set $ResultSet_{as}$.

3.1.5 Discussion

As a difference to previous work [91], we dropped the point-centric view of dimension qualification and measure application. We replaced the point-centric view in favour of the set-oriented view of SQL that leads to a clearer understanding for users. The notion of “concept” that belongs to an ontology-based view is replaced by the notion of “predicates”. Measures are defined independently from cubes to increase re-usability, although they can be linked to cubes. Semantic enrichment of OLAP cubes by multi-dimensional ontologies can be found in [95].

These design decisions are based on the experience gathered from the case studies. Users are more common with a “template-oriented” view of non-comparative analysis situations than with the ontology-based view as presented in [91]. A user can fill in familiar items to specify conceptually the query she or he needs.

Another difference to [91] represents the omission of multi-dimensional predicates (corresponding to multi-dimensional concepts in [91]). For instance, in APMN4BI, one cannot define a single predicate that expresses “rural insurants treated by urban doctors”. Such predicates would have to be defined over more than one dimension. Of course, non-comparative analysis situations (and the underlying eDFM) could be extended by such additional constructs. But the case studies showed that such overload is not necessary. By simply selecting appropriate predicates in each dimension (*InsInRuralDistrict* and *DocInUrbanDistrict*), the non-comparative analysis

situation expresses the user intention as well as one predicate that combines both.

Note, that in the SQL translation, we only use inner joins. This corresponds to common practice of using star schemas as implementation of DFMs. If users want to select, for example, all insurants with and without drug prescriptions, the data has to be prepared appropriately, although extensions to non-comparative analysis situations could be adopted to also allow left and right outer joins.

3.2 Comparative Analysis Situations

Comparison is a main activity of a business analyst. She or he is interested in an analysis situation and wants to compare it with another one to obtain assistance for decision making, e.g., compare drug prescription costs per province with those of the previous year. Comparisons are expressed by comparative analysis situations that can be considered as a join of result sets of two non-comparative analysis situations with different contexts (context of interest and context of comparison). For instance, the context of interest corresponds to the actual year and the context of comparison to the previous year. The result of a comparison itself is expressed by a score, for example, ratio of drug prescription costs of a year to drug prescription costs of the previous year. The result set of a comparative analysis situation comprises both the result set of the context of interest and the joined result set of the context of comparison, and, additionally, one or more scores expressing the comparison result. Optionally, this query result can be filtered depending on the score value.

3.2.1 Preliminary Remarks

Join conditions were introduced in Subsection 2.4.1.1 of Chapter 2 as constituents of an eDFM. A join condition represents a boolean expression defined over qualified dimension levels qualified by CoI and CoC, and possibly combined with dimensional operators. Join conditions are defined at schema

level. The application of join conditions at instance level links a result set of the context of interest and a result set of the context of comparison. The name and the expression of a join condition are returned by the functions *NameOf* and *ExprOf*. A join condition describes how dimension nodes of the context of interest are combined with dimension nodes of the context of comparison. The example of an eDFM in Chapter 2.4, contains two join conditions named as *SameInsProvince* and *PrevYear*, and defined by boolean expressions $\text{CoI.insProvince} = \text{CoC.insProvince}$ and $\text{prevYear}(\text{CoI.year}) = \text{CoC.year}$. One could define further join conditions using boolean expressions with, for instance, other dimensional operators or other comparison operators. As an example, by boolean expression $\text{monthOfPrevYear}(\text{CoI.month}) = \text{CoC.month}$ a month is joined with the month of the previous year or, as another example, boolean expression $\text{CoI.year} > \text{CoC.year}$ joins a year with all preceding years. For later use in this section, we assign names *SameMonth* and *AllPrecYears*, respectively, to these join conditions, i.e., *SameMonth* $\stackrel{\text{def}}{=} \text{monthOfPrevYear}(\text{CoI.month}) = \text{CoC.month}$ and *AllPrecYears* $\stackrel{\text{def}}{=} \text{CoI.year} > \text{CoC.year}$. Moreover, in the following subsection, another join condition is used: *SameMonthName* $\stackrel{\text{def}}{=} \text{monthName}(\text{CoI.month}) = \text{monthName}(\text{CoC.month})$. This join condition comprises a new dimensional operator *monthName* at level *month* of dimension schema *Time* (not depicted in Figure 2.3) that returns the name of a dimension node of level *month*, for example, $\text{monthName}(\text{May 2016})$ returns name *May* of dimension node *May 2016*.

Scores and score predicates were introduced in Subsection 2.4.2.5 and Subsection 2.4.2.6 of Chapter 2 as enrichments of a dimensional fact model. A score is used to assess a non-comparative analysis situation (context of interest) with respect to another one (context of comparison). Aggregate measures of both analysis situations are combined to express the difference of both. Scores can be used for additional filtering of result sets of comparative queries by applying score predicates. Scores and score predicates obtain names and are defined by arithmetic and boolean expressions, respectively. The name and the expression of a score and score predicate are returned by the functions *NameOf* and *ExprOf*. In the example of Chapter 2.4, the eDFM con-

tains two scores *RatioOfSumOfCosts* and *RatioOfAvgCostsPerInsurant* defined by the expressions $\text{CoI}.\text{SumOfCosts} / \text{CoC}.\text{SumOfCosts}$ and $\text{CoI}.\text{AvgCostsPerInsurant} / \text{CoC}.\text{AvgCostsPerInsurant}$. Whereas these scores evaluate the ratios of costs, one could also define, for example, a score with name *PercDiffOfSumOfCosts* and arithmetic expression $(\text{CoI}.\text{SumOfCosts} - \text{CoC}.\text{SumOfCosts}) / \text{CoC}.\text{SumOfCosts} * 100$ which represents a relative cost increase as a percentage. A score with name *DiffOfAvgCosts* and defining expression $\text{CoI}.\text{AvgCosts} - \text{CoC}.\text{AvgCosts}$ based on the definition of aggregate measure $\text{AvgCosts} \stackrel{\text{def}}{=} \text{AVG}(\text{costs})$ would be another example. This score defines an absolute difference of average costs.

The example of an eDFM in Chapter 2 presents one score predicate with name *IncreasedAvgCostsPerInsurant* and defining boolean expression $\text{RatioOfAvgCostsPerInsurant} > 1$ that can be used to filter rows of a comparative query having increased average costs per insurant. More complex score predicates can be defined, for instance, a score predicate with name *CostsOutOfStdDev* that examines whether score *DiffOfAvgCosts* lies out of the interval enclosed by standard deviation *StdDevCosts* which can be defined as another aggregate measure $\text{StdDevCosts} \stackrel{\text{def}}{=} \text{STDDEV}(\text{costs})$. In this case, score predicate *CostsOutOfStdDev* can be defined by boolean expression $\text{DiffOfAvgCosts} < (-1) * \text{CoC}.\text{StdDevCosts} \text{ OR } \text{DiffOfAvgCosts} > \text{CoC}.\text{StdDevCosts}$. In this boolean expression, also arithmetic expressions and aggregate measures are used additionally to scores, comparative operators, and logical operators.

In the subsequent subsections, a formal definition of a comparative analysis situation is presented that uses join conditions, scores, and score predicates as additional constituents. Thereafter, graphical representations and a translation into SQL queries are given. The translation into SQL defines the semantics of comparative analysis situations.

3.2.2 Formal Definition

A formal definition of a comparative analysis situation is introduced in this subsection. Two non-comparative analysis situations are compared in a way

that one represents the context of interest and the other one the context of comparison. In our definition, both non-comparative analysis situations can be grounded on different cube instance of the same cube schema.⁵ Dimension nodes of both non-comparative analysis situations are linked by join conditions. Scores are used as a measure of comparison and score predicates are used as score filters to select rows of the query result that exhibit specific score values.

Definition 3.4. A *comparative analysis situation* $\mathbf{cas} = (\mathbf{as}^I, \mathbf{as}^C, J, S, SF)$ comprises

1. a non-comparative analysis situation \mathbf{as}^I as *context of interest* (abbreviated as *CoI*),
2. a non-comparative analysis situation \mathbf{as}^C as *context of comparison* (abbreviated as *CoC*) such that $CubeSchema_{\mathbf{as}^I} = CubeSchema_{\mathbf{as}^C}$,
3. a set J of join conditions defined over qualified dimension levels of dimension schemas in $DimSchemas_{\mathbf{as}^I} (= DimSchemas_{\mathbf{as}^C})$ ⁶ qualified by *CoI* and *CoC* such that dimension levels qualified by *CoI* refer to \mathbf{as}^I and dimension levels qualified by *CoC* refer to \mathbf{as}^C ,
4. a set S of scores defined over qualified aggregate measures in $AMsrs_{\mathbf{as}^I}$ and $AMsrs_{\mathbf{as}^C}$ qualified by *CoI* and *CoC* such that aggregate measures qualified by *CoI* refer to \mathbf{as}^I and aggregate measures qualified by *CoC* refer to \mathbf{as}^C , and
5. a set SF of score predicates (also denoted as *score filters*) defined over S , $AMsrs_{\mathbf{as}^I}$, and $AMsrs_{\mathbf{as}^C}$.

Moreover, in the context of \mathbf{cas} , we define $CoI_{\mathbf{cas}} = \mathbf{as}^I$, $CoC_{\mathbf{cas}} = \mathbf{as}^C$, $JoinConds_{\mathbf{cas}} = J$, $Scores_{\mathbf{cas}} = S$, and $ScoreFilters_{\mathbf{cas}} = SF$. \square

⁵In most cases, we also use the same cube instance for both the context of interest and the context of comparison. There might be reasons to compare different cube instances. For example, one wants to compare drug prescription costs of one insurer with drug prescription costs of another insurer that are stored in two different cubes. But note that both cube instances must be of the same cube schema.

⁶Note that the same cube schema is used for both the context of interest and the context of comparison. Thus, both contexts also use the same dimension schemas.

In the following example, a business analyst would like to compare high drug prescription costs of the year 2016 with the previous year 2015. She or he wants to detect increases of high drug prescription costs in rural insurants' districts in year 2016 compared with previous year 2015 listed per insurants' province. We specify a comparative analysis situation $\mathbf{cas} = \text{HighDrugPrescrCostsInRuralDistrictsComparedWithPrevYear}$. First, we define the context of interest $CoI_{\mathbf{cas}}$ which refers to drug prescription costs of year 2016:

- $CubeInstance_{CoI_{\mathbf{cas}}} = \text{DrugPrescription}$
- $BMsrConds_{CoI_{\mathbf{cas}}} = \{\text{HighCostsPerUnit}\}$
- $AMsrs_{CoI_{\mathbf{cas}}} = \{\text{SumOfCosts}, \text{AvgCostsPerInsurant}\}$
- $DimQuals_{CoI_{\mathbf{cas}}} = \{(\text{Time}, \text{year}, 2016, \emptyset, \text{top}), (\text{Insurant}, \text{top}, \text{all}, \{\text{InsInRuralDistrict}\}, \text{insProvince}), (\text{Drug}, \text{top}, \text{all}, \emptyset, \text{top}), (\text{Doctor}, \text{top}, \text{all}, \emptyset, \text{top}),\}$
- $FilterConds_{CoI_{\mathbf{cas}}} = \{\text{HighAvgDrugPrescrCostsPerIns}\}$

The context of interest $CoI_{\mathbf{cas}}$ is compared with the context of comparison $CoC_{\mathbf{cas}}$ that refers to year 2015 and which is defined in the following way:

- $CubeInstance_{CoC_{\mathbf{cas}}} = \text{DrugPrescription}$
- $BMsrConds_{CoC_{\mathbf{cas}}} = \{\text{HighCostsPerUnit}\}$
- $AMsrs_{CoC_{\mathbf{cas}}} = \{\text{SumOfCosts}, \text{AvgCostsPerInsurant}\}$
- $DimQuals_{CoC_{\mathbf{cas}}} = \{(\text{Time}, \text{year}, 2015, \emptyset, \text{top}), (\text{Insurant}, \text{top}, \text{all}, \{\text{InsInRuralDistrict}\}, \text{insProvince}), (\text{Drug}, \text{top}, \text{all}, \emptyset, \text{top}), (\text{Doctor}, \text{top}, \text{all}, \emptyset, \text{top}),\}$
- $FilterConds_{CoC_{\mathbf{cas}}} = \{ \}$

Note, that CoC_{cas} refers to year 2015 and has no filter condition compared with CoI_{cas} . CoC_{cas} is not restricted to high average drug prescription costs per insurant because high average drug prescription costs per insurant in 2016 must be joined with drug prescription costs per insurant in 2015 at level province, independently whether these costs were high in 2015 or not.

Finally, the set of join conditions, the set of scores, and the set of score filters are defined:

- $JoinConds_{\text{cas}} = \{SameInsProvince\}$
- $Scores_{\text{cas}} = \{RatioOfSumOfCosts, RatioOfAvgCostsPerInsurant\}$
- $ScoreFilters_{\text{cas}} = \{IncreasedAvgCostsPerInsurant\}$

Set $JoinConds_{\text{cas}}$ contains only one join condition that expresses that comparison is done per insurants' province. No further join conditions are necessary because the dimension qualification in dimension schema *Insurant* is the only one having a granularity level unequal to the top level. The purpose of join condition *SameInsProvince* is to only compare drug prescription costs of the same province. Scores *RatioOfSumOfCosts* and *RatioOfAvgCostsPerInsurant* are computed and listed per province. Finally, only those rows are elements of the result set of the comparative analysis situation that have increased costs per insurant that is obtained by score filter *IncreasedAvgCostsPerInsurant*.

If one likes to refine comparison to months, she or he can add join condition *SameMonthName* as introduced in Subsection 3.2.1.⁷ The following constituents have to be adapted:

- $DimQual_{CoI_{\text{cas}}}(\text{Time}) = (\text{Time}, \text{year}, 2016, \emptyset, \text{month})$
- $DimQual_{CoC_{\text{cas}}}(\text{Time}) = (\text{Time}, \text{year}, 2015, \emptyset, \text{month})$
- $JoinConds_{\text{cas}} = \{SameInsProvince, SameMonthName\}$

⁷In this case, we use join condition *SameMonthName* instead of *SameMonth*. By join condition *SameMonth*, only records are joined that have the same month and year. In contrast, by join condition *SameMonthName*, the join condition only compares the month name independently from the year. Thus, months of arbitrary years can be compared.

Another variation of this example would be, if one wants to compare drug prescription costs of all preceding years. In this case, we have to make the following adaptations where we use join condition *AllPrecYears* defined in Subsection 3.2.1:

- $DimQual_{CoCas}(Time) = (Time, top, all, \emptyset, year)$
- $JoinConds_{cas} = \{SameInsProvince, AllPrecYears\}$

In another modified example, score and score filters could be changed. For instance, one would like to examine whether the difference of the average drug prescription costs lies out of the standard deviation. The following modifications have to be done to express this comparison. We use aggregate measure *AvgCosts*, score *DiffOfAvgCosts*, and score predicate *CostsOutOfStdDev* introduced in Subsection 3.2.1:

- $AMsrs_{CoIcas} = \{SumOfCosts, AvgCostsPerInsurant, AvgCosts\}$
- $AMsrs_{CoCas} = \{SumOfCosts, AvgCostsPerInsurant, AvgCosts\}$
- $Scores_{cas} = \{RatioOfSumOfCosts, RatioOfAvgCostsPerInsurant, DiffOfAvgCosts\}$
- $ScoreFilters_{cas} = \{CostsOutOfStdDev\}$

This variations demonstrate further slightly adapted examples. In the following subsections, we focus on the original version of this example.

3.2.3 Graphical Representation

This subsection describes graphical representations of comparative analysis situations. As for non-comparative analysis situations, one representation (Figure 3.11 and Figure 3.12) has an obvious one-to-one correspondence to the formal definition (*full representation*), the second representation (Figure 3.13 and Figure 3.14) makes some visual simplifications but also sets a focus on the bare essentials (*lean representation*), and the third one (Figure 3.15

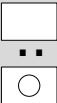






 cas description of the comparative analysis situation	
 Context of Interest (CoI)	 Context of Comparison (CoC)
Cube: c  BMSR-Filter: $\{ B_1, \dots, B_b \}$  Measures: $\{ M_1, \dots, M_m \}$	Cube: c'  BMSR-Filter: $\{ B'_1, \dots, B'_b \}$  Measures: $\{ M'_1, \dots, M'_m \}$
Dimension Qualification	
D₁ DL: L_1 DN: N_1 SC: $\{ P_{1,1}, \dots, P_{1,s1} \}$ GL: G_1	D₁ DL: L'_1 DN: N'_1 SC: $\{ P'_{1,1}, \dots, P'_{1,s'1} \}$ GL: G'_1
D_n DL: L_n DN: N_n SC: $\{ P_{n,1}, \dots, P_{n,sn} \}$ GL: G_n	D_n DL: L'_n DN: N'_n SC: $\{ P'_{n,1}, \dots, P'_{n,sn} \}$ GL: G'_n
Filter: $\{ F_1, \dots, F_k \}$	
Join Condition: $\{ J_1, \dots, J_t \}$	
Scores: $\{ S_1, \dots, S_v \}$	
Score Filter: $\{ F^o_1, \dots, F^o_w \}$	

Figure 3.11: Template of a comparative analysis situation depicted in full graphical representation

and Figure 3.16) is a condensed form that hides details and that can only be used in combination with navigation operators (*condensed representation*).

Figure 3.11 shows a formal template of the full graphical representation of a comparative analysis situation. The first section of the graphical representation comprises the name (in the formal template denoted as **cas**), an optional description not presented in the formal definition, and a symbol which associates comparison. For each constituent of a comparative analysis situation, one can see a pictogram. The context of interest and the context of comparison are represented by two non-comparative analysis situations. Instead of names for them, we have headings **Context of Interest (Col)** and **Context of Comparison (CoC)** (or shortly written as **Col** and **CoC**). The pictogram of **Col** is drawn as a rectangle with a circle, the pictogram for **CoC** as a rectangle used for non-comparative analysis situations. The whole construct of a comparative analysis situation obtains its own pictogram (the symbol for **Col** followed by a colon and the symbol for **CoC** in the first section of the graphical representation), too. Additional constituents for comparative analysis situations (set of join conditions, set of scores, and set of score filters) that are not parts of **Col** and **CoC** are represented by separate symbols and labels. The pictograms for join conditions, scores, and score filters are taken from the symbols of an eDFM that contains their definitions.

All constituents of the formal definition of a comparative analysis situation can be found in the full graphical representation. Constituents of the context of comparison have a “prime” to distinguish them from constituents of the context of interest. Moreover, $CubeInstance_{CoI_{cas}} = \mathbf{c}$, $AMsrs_{CoI_{cas}} = \{M_1, \dots, M_m\}$, $BMsrPredicates_{CoI_{cas}} = \{B_1, \dots, B_b\}$, $FilterConds_{CoI_{cas}} = \{F_1, \dots, F_k\}$, $DimSchemas_{CoI_{cas}} = \{D_1, \dots, D_n\}$, and, for $1 \leq i \leq n$, $DiceLvl_{CoI_{cas}}(D_i) = L_i$, $DiceNode_{CoI_{cas}}(D_i) = N_i$, $SliceConds_{CoI_{cas}}(D_i) = \{P_{i,1}, \dots, P_{i,s_i}\}$, and $GranLvl_{CoI_{cas}}(D_i) = G_i$ are constituents of the context of interest. $CubeInstance_{CoC_{cas}} = \mathbf{c}'$ (with $CubeSchema_{CoI_{cas}} = CubeSchema_{CoC_{cas}}$), $AMsrs_{CoC_{cas}} = \{M'_1, \dots, M'_{m'}\}$, $BMsrPredicates_{CoC_{cas}} = \{B'_1, \dots, B'_{b'}\}$, $FilterConds_{CoC_{cas}} = \{F'_1, \dots, F'_{k'}\}$, $DimSchemas_{CoC_{cas}} = \{D_1, \dots, D_n\}$ ($= DimSchemas_{CoI_{cas}}$), and, for $1 \leq i \leq n$, $DiceLvl_{CoC_{cas}}(D_i) = L'_i$, $DiceNode_{CoC_{cas}}(D_i) = N'_i$, $SliceConds_{CoC_{cas}}(D_i) = \{P'_{i,1}, \dots, P'_{i,s'_i}\}$, and

$GranLvl_{CoCas}(D_i) = G'_i$ belong to the context of comparison. $JoinConds_{cas} = \{J_1, \dots, J_t\}$, $Scores_{cas} = \{S_1, \dots, S_v\}$, and $ScoreFilters_{cas} = \{F_1^\circ, \dots, F_w^\circ\}$ are further constituents of the comparative analysis situation.

Similar to conditions of non-comparative analysis situations, the join conditions and score filters result in the overall join condition $J_1 \wedge \dots \wedge J_t$ and overall score filter $F_1^\circ \wedge \dots \wedge F_w^\circ$. If there are no join conditions, scores, or score filters, we write an empty set respectively.

Figure 3.12 demonstrates an example of a full graphical representation of a comparative analysis situation specified in 3.2.2. Both non-comparative analysis situations (**Col** and **CoC**) also contain default values: *top* as dice and granularity level, *all* as dice node, and the empty set as slice and filter condition.

In Figure 3.13, a template for a lean graphical notation is depicted. As for non-comparative analysis situations, this graphical representation focuses on the substantial information and allows more flexibility in drawing. But, again, all necessary information is contained such that this representation can be transformed in a one-to-one correspondence to a full graphical notation. The context of interest and the context of comparison are depicted within separate rounded rectangles which are included in the outer frame. The outer rounded rectangle also contains additional elements of a comparative analysis situation: join conditions, scores, and score filters. Again, labels are omitted and only pictograms are used as a graphical prefix beside the constituents and, in the case of sets, there is no usual set notation but each set element is prefixed by the corresponding pictogram. As for non-comparative analysis situations, all constituents can be positioned arbitrarily respecting the membership relation via nested rounded rectangles. Default values *top*, *all*, and the empty set are not stated but they can be considered as implicitly present. Each information of the formal definition of a comparative analysis situation can be derived from the lean graphical representation.

The lean graphical notation of the example of Figure 3.12 can be found in Figure 3.14. As for non-comparative analysis situations, only substantial information is included and default values are omitted. The graphical constituents are positioned in a freestyle to obtain a compact graphical rep-

<input type="checkbox"/> : <input type="checkbox"/> HighDrugPrescrCostsInRuralDistrictsComparedWithPrevYear Increases of high drug prescription costs in rural insurers' districts in year 2016 compared with previous year 2015 listed per insurers' province	
<input type="checkbox"/>	Context of Comparison (CoC)
<input type="checkbox"/>	Context of Interest (CoI)
<input type="checkbox"/>	Cube: DrugPrescription BMSr-Filter: { HighCostsPerUnit } Measures: { SumOfCosts, AvgCostsPerInsurant }
Dimension Qualification	Dimension Qualification
Time DL: year DN: 2016 SC: { } GL: top	Insurant DL: top DN: all SC: { InsInRuralDistrict } GL: insProvince
Drug DL: top DN: all SC: { } GL: top	Doctor DL: top DN: all SC: { } GL: top
Filter: { HighAvgDrugPrescrCostsPerIns }	Filter: { }
Join Condition: { SameInsProvince }	
Scores: { RatioOfSumOfCosts, RatioOfAvgCostsPerInsurant }	
Score Filter: { IncreasedAvgCostsPerInsurant }	

Figure 3.12: Example of a comparative analysis situation depicted in full graphical representation

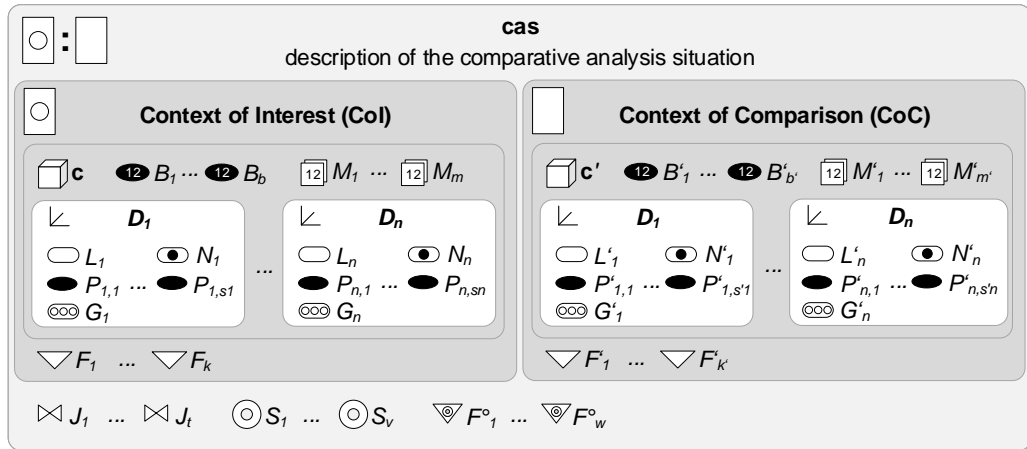


Figure 3.13: Template of a comparative analysis situation depicted in lean graphical representation

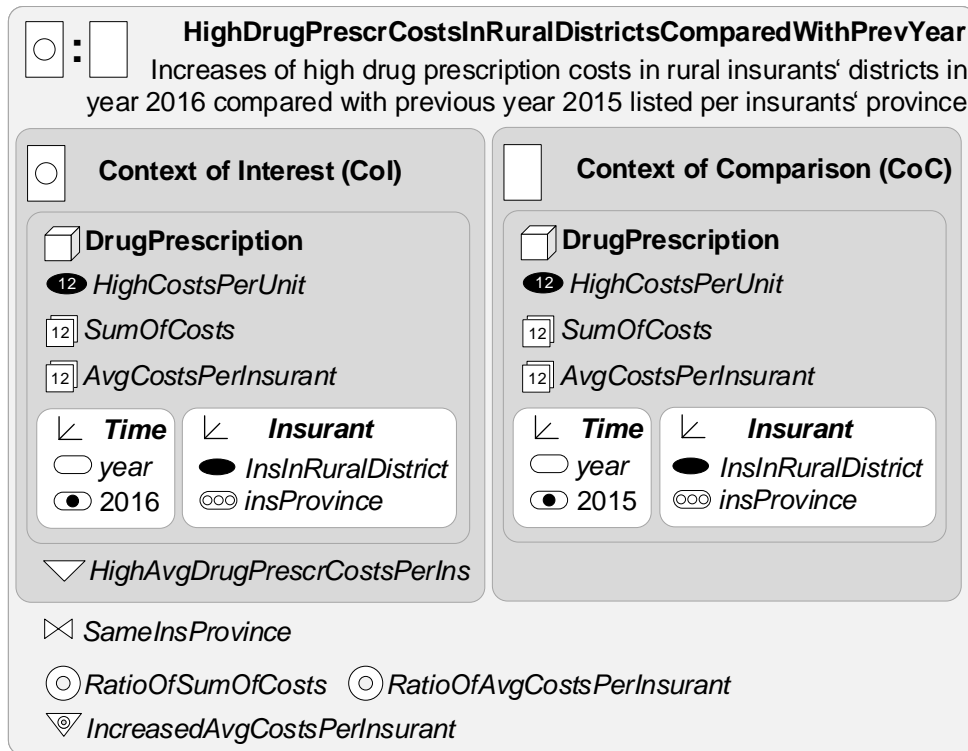


Figure 3.14: Example of a comparative analysis situation depicted in lean graphical representation

resentation.

A formal template of a condensed graphical representation is shown in Figure 3.15 and Figure 3.16 presents an example associated with the example of Figure 3.12. The condensed representation does not contain details of the comparative analysis situation. It only contains the name and an optional description of the comparative analysis situation. Additional information is needed to transform the condensed representation into the graphical notation of Figure 3.12 or Figure 3.14. As for non-comparative analysis situations, this condensed shape only can be used in combination with navigation operators.

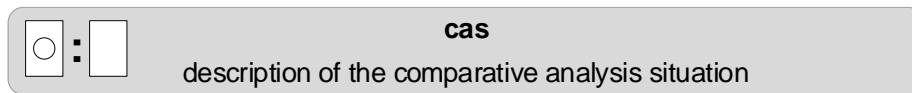


Figure 3.15: Template of a comparative analysis situation depicted in condensed graphical representation

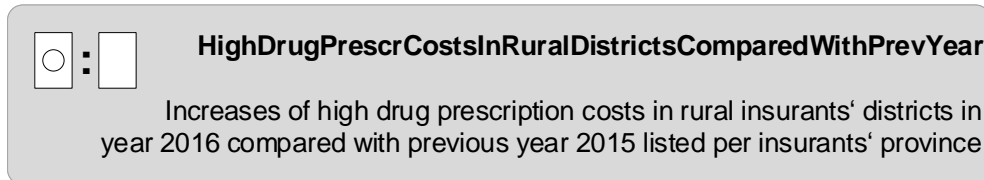


Figure 3.16: Example of a comparative analysis situation depicted in condensed graphical representation

3.2.4 Translation into SQL

Like in the case of non-comparative analysis situations, we define the semantics of a comparative analysis situation by a mapping into an SQL query. The context of interest and the context of comparison are translated into SQL queries of non-comparative analysis situations. The result sets of both are joined via an outer query which also computes scores and optionally applies score filters. Again, this translation is based on a relational star schema.

Definition 3.5. Let **cas** be a comparative analysis situation such that the constituents of the context of interest are defined by $CubeInstance_{CoI_{\mathbf{cas}}} = \mathbf{c}$, $AMsrs_{CoI_{\mathbf{cas}}} = \{M_1, \dots, M_m\}$, $BMsrPredicates_{CoI_{\mathbf{cas}}} = \{B_1, \dots, B_b\}$, $FilterConds_{CoI_{\mathbf{cas}}} = \{F_1, \dots, F_k\}$, $DimSchemas_{CoI_{\mathbf{cas}}} = \{D_1, \dots, D_n\}$, $DimInstances_{\mathbf{c}} = \{\mathbf{d}_1, \dots, \mathbf{d}_n\}$, and, for $1 \leq i \leq n$, $DimSchema_{\mathbf{d}_i} = D_i$, $DiceLvl_{CoI_{\mathbf{cas}}}(D_i) = L_i$, $DiceNode_{CoI_{\mathbf{cas}}}(D_i) = N_i$, $SliceConds_{CoI_{\mathbf{cas}}}(D_i) = \{P_{i,1}, \dots, P_{i,s_i}\}$, and $GranLvl_{CoI_{\mathbf{cas}}}(D_i) = G_i$. Furthermore, the constituents of the context of comparison are defined by $CubeInstance_{CoC_{\mathbf{cas}}} = \mathbf{c}'$, $AMsrs_{CoC_{\mathbf{cas}}} = \{M'_1, \dots, M'_{m'}\}$, $BMsrPredicates_{CoC_{\mathbf{cas}}} = \{B'_1, \dots, B'_{b'}\}$, $FilterConds_{CoC_{\mathbf{cas}}} = \{F'_1, \dots, F'_{k'}\}$, $DimSchemas_{CoC_{\mathbf{cas}}} = \{D_1, \dots, D_n\}$ ($= DimSchemas_{CoI_{\mathbf{cas}}}$), $DimInstances_{\mathbf{c}'} = \{\mathbf{d}'_1, \dots, \mathbf{d}'_n\}$, and, for $1 \leq i \leq n$, $DimSchema_{\mathbf{d}'_i} = D_i$, $DiceLvl_{CoC_{\mathbf{cas}}}(D_i) = L'_i$, $DiceNode_{CoC_{\mathbf{cas}}}(D_i) = N'_i$, $SliceConds_{CoC_{\mathbf{cas}}}(D_i) = \{P'_{i,1}, \dots, P'_{i,s'_i}\}$, and $GranLvl_{CoC_{\mathbf{cas}}}(D_i) = G'_i$. Finally, join conditions, scores, and score filters are defined by $JoinConds_{\mathbf{cas}} = \{J_1, \dots, J_t\}$, $Scores_{\mathbf{cas}} = \{S_1, \dots, S_v\}$, and $ScoreFilters_{\mathbf{cas}} = \{F_1^\circ, \dots, F_w^\circ\}$.

The *semantics* of **cas** is denoted by $Query_{\mathbf{cas}}$ and is defined by an SQL query as presented in the template of Figure 3.17 that is based on the relational schemas of cube instances \mathbf{c} and \mathbf{c}' .

The result set of an execution of the SQL query that defines the semantics of a comparative analysis situation **cas** is denoted by $ResultSet_{\mathbf{cas}}$. The relational schema of result set $ResultSet_{\mathbf{cas}}$ is denoted by $ResultSetSchema_{\mathbf{cas}}$ and comprises the following attributes: $CoI_NameOf(G_1), \dots, CoI_NameOf(G_n), CoI_NameOf(M_1), \dots, CoI_NameOf(M_m), CoC_NameOf(G'_1), \dots, CoC_NameOf(G'_n), CoC_NameOf(M'_1), \dots, CoC_NameOf(M'_{m'}), NameOf(S_1), \dots, NameOf(S_v)$. \square

Figure 3.17 defines the translation of a comparative analysis situation **cas** into an SQL query. The query comprises two sub-queries that are embedded in an outer query. Both sub-queries are translations of the non-comparative analysis situations representing the context of interest and the context of comparison, and they obtain alias names **CoI** and **CoC**. The translations of these sub-queries correspond to the semantics of non-comparative analysis situations which is specified in Definition 3.3. This definition assumes that

```

SELECT      CoI.NameOf(G1) AS CoI_NameOf(G1), ...,
            CoI.NameOf(Gn) AS CoI_NameOf(Gn),
            CoI.NameOf(M1) AS CoI_NameOf(M1), ...,
            CoI.NameOf(Mm) AS CoI_NameOf(Mm),
            CoC.NameOf(G'1) AS CoC_NameOf(G'1), ...,
            CoC.NameOf(G'n) AS CoC_NameOf(G'n),
            CoC.NameOf(M'1) AS CoC_NameOf(M'1), ...,
            CoC.NameOf(M'm') AS CoC_NameOf(M'm'),
            ExprOf(S1) AS NameOf(S1), ..., ExprOf(Sv) AS NameOf(Sv)
FROM        ( SELECT NameOf(G1), ..., NameOf(Gn),
                    ExprOf(M1) AS NameOf(M1), ...,
                    ExprOf(Mm) AS NameOf(Mm)
            FROM      NameOf(c) NATURAL JOIN
                    NameOf(d1) NATURAL JOIN ... NATURAL JOIN NameOf(dn)
            WHERE     ExprOf(B1) AND ... AND ExprOf(Bb) AND
                    NameOf(L1) = ExprOf(N1) AND ... AND
                    NameOf(Ln) = ExprOf(Nn) AND
                    ExprOf(P1,1) AND ... AND ExprOf(P1,s1) AND ... AND
                    ExprOf(Pn,1) AND ... AND ExprOf(Pn,sn)
            GROUP BY NameOf(G1), ..., NameOf(Gn)
            HAVING    ExprOf(F1) AND ... AND ExprOf(Fk)
                    ) CoI
INNER JOIN  ( SELECT NameOf(G'1), ..., NameOf(G'n),
                    ExprOf(M'1) AS NameOf(M'1), ...,
                    ExprOf(M'm') AS NameOf(M'm')
            FROM      NameOf(c') NATURAL JOIN
                    NameOf(d'1) NATURAL JOIN ... NATURAL JOIN NameOf(d'n)
            WHERE     ExprOf(B'1) AND ... AND ExprOf(B'b') AND
                    NameOf(L'1) = ExprOf(N'1) AND ... AND
                    NameOf(L'n') = ExprOf(N'n') AND
                    ExprOf(P'1,1) AND ... AND ExprOf(P'1,s'1) AND ... AND
                    ExprOf(P'n,1) AND ... AND ExprOf(P'n,s'n)
            GROUP BY NameOf(G'1), ..., NameOf(G'n)
            HAVING    ExprOf(F'1) AND ... AND ExprOf(F'k')
                    ) CoC
ON          ExprOf(J1) AND ... AND ExprOf(Jt)
WHERE       ExprOf(F1o) AND ... AND ExprOf(Fwo)

```

Additional translation rules: (1) Additional translation rules of non-comparative analysis situations are applied on the sub-select-statements of CoI and CoC (see Definition 3.3). (2) An empty ON and WHERE clause is removed from the outer SQL statement.

Figure 3.17: Formal specification of the translation of a comparative analysis situation into SQL

cube instances \mathbf{c} and \mathbf{c}' are organized in star schemas as relational database tables and that the naming conventions for attributes allow to use natural joins. The outer SQL statement comprises join conditions in the **ON**-clause with respect to the sub-queries and optional score filters in the **WHERE**-clause. In the case that join conditions or score filters are missing, the **ON**-clause and **WHERE**-clause, respectively, are omitted which is expressed by the second additional translation rule listed in Figure 3.17. The **SELECT**-clause of the outer SQL statement contains the granularity levels and aggregate measures of the context of interest and the context of comparison. In both cases, the selected attributes of the corresponding sub-queries are used with alias names **CoI** and **CoC**, and the column names are prefixed by *CoI_* and *CoC_* to distinguish names for granularity levels and aggregate measures of the context of interest and the context of comparison. Additionally, the outer **SELECT**-clause comprises the computation of scores and the score names are used as alias names. In the whole SQL statement, functions *NameOf* and *ExprOf* introduced in the previous chapter are used at a meta level to obtain appropriate names and resolved expressions for constructing correct SQL statements.⁸ Aggregate measures, base measure conditions, slice conditions, and filter conditions of the context of interest and context of comparison, and, join conditions, scores, and score filters are resolved with respect to their defining expressions. For all these constituents, we assume that these expressions are conform with SQL syntax.

For illustration only, a template of a result set of a comparative analysis situation is depicted in Figure 3.18. The names of the granularity levels G_1, \dots, G_n and the names of the aggregate measures M_1, \dots, M_m of the context of interest, and the names of the granularity levels G'_1, \dots, G'_n and the names of the aggregate measures $M'_1, \dots, M'_{m'}$ of the context of comparison are used as column names of the result table. These column names have to be additionally prefixed by *CoI_* and *CoC_* that leads to column names *CoI_NameOf*(G_1), \dots , *CoI_NameOf*(G_n), *CoI_NameOf*(M_1), \dots ,

⁸Note that for the generation of column names, a prefix (written in *slant* font) is concatenated with the string value returned by function *NameOf*. For instance, expression *CoI_NameOf*(*insProvince*) represents column name *CoIinsProvince*.

$CoL_NameOf(G_1)$...	$CoL_NameOf(G_n)$	$CoL_NameOf(M_1)$...	$CoL_NameOf(M_m)$	$CoC_NameOf(G'_1)$...	$CoC_NameOf(G'_n)$	$CoC_NameOf(M'_1)$...	$CoC_NameOf(M'_{m'})$	$NameOf(S_1)$...	$NameOf(S_v)$
...
...
...

Figure 3.18: Relational schema of a result set of a comparative analysis situation

$CoL_NameOf(M_m)$, $CoC_NameOf(G'_1)$, \dots , $CoC_NameOf(G'_n)$, $CoC_NameOf(M'_1)$, \dots , $CoC_NameOf(M'_{m'})$. Finally, there is a column for each score such that score names S_1, \dots, S_v are also used as column names.

Figure 3.19 demonstrates an SQL translation of comparative analysis situation presented in Figure 3.12. The queries for the context of interest (associated with alias name **CoI**) and the context of comparison (associated with alias name **CoC**) are applied to instances of the eDFM of Figure 2.3. They are constructed via a natural join on fact table **DrugPrescription**, and dimension tables **Time** and **Insurant**. Aggregate measures *SumOfCosts* and *AvgCostsPerInsurant* are computed and returned per insurants' province accordingly to granularity level *insProvince* which can be found in the **SELECT**- and **GROUP-BY**-clause. Rows from fact table **DrugPrescription** are restricted with respect to base measure condition *HighCostsPerUnit* resolved to expression $costs / quantity > 50$. Additionally, slice condition *InsInRuralDistrict* is applied to both sub-queries which is translated to expression $inhPerSq-kmInInsDistr < 400$. The difference of sub-query **CoI** and **CoC** lies in the translation of the dice node and the translation of the filter condition for aggregate measures. In the sub-query for **CoI**, the **WHERE**-clause contains restriction $year = 2016$ whereas for **CoC**, condition $year = 2015$ is stated. Sub-query **CoI** has an additional **HAVING**-clause with expression $SUM(costs) / COUNT(DISTINCT insurant) > 1000$ that implements filter condition *High-AvgDrugPrescrCostsPerIns*. In sub-query **CoI** no **HAVING**-clause can be found

because there is no filter condition at all. The result of both sub-queries is combined by an inner join of the outer select statement such that join condition *SameInsProvince* is translated into an **ON**-clause containing the defining expression $\text{CoI.insProvince} = \text{CoC.insProvince}$. The **SELECT**-clause of the outer SQL statement comprises the granularity levels and the aggregate measures transferred from the sub-queries for **CoI** and **CoC**. Additionally, it also contains expressions for scores ($\text{CoI.SumOfCosts} / \text{CoC.SumOfCosts}$ and $\text{CoI.AvgCostsPerInsurant} / \text{CoC.AvgCostsPerInsurant}$) aliased by score names *RatioOfSumOfCosts* and *RatioOfAvgCostsPerInsurant*. In the **WHERE**-clause of the outer select statement, the expression $\text{CoI.AvgCostsPerInsurant} / \text{CoC.AvgCostsPerInsurant} > 1$ for the translated score filter *HighAvgDrugPrescrCostsPerIns* can be found.

For illustration, Figure 3.20 shows a possible result table with fictitious data. This table contains columns *CoI.insProvince* and *CoC.insProvince* for granularity levels, and columns *CoI.SumOfCosts*, *CoC.SumOfCosts*, *CoI.AvgCostsPerInsurant*, and *CoC.AvgCostsPerInsurant* for aggregate measures with respect to the context of interest (prefixed by *CoI*) and the context of comparison (prefixed by *CoC*). Columns *RatioOfSumOfCosts* and *RatioOfAvgCostsPerInsurant* comprise score values. Due to join condition *SameInsProvince* defined by expression $\text{CoI.insProvince} = \text{CoC.insProvince}$, in this example, the values of columns *CoI.insProvince* and *CoC.insProvince* are equal in each result row.

3.2.5 Discussion

As for non-comparative analysis situations, we also reduced the ontology-based approach as presented in [91]. We do not use notions like comparative facts, comparative cubes and comparative concepts. The point-centric view is omitted. Instead of point of interest and point of comparison, we use the notions context of interest and context of comparison. Both contexts are considered as non-comparative analysis situations for which their results are joined via join conditions that are predefined at schema level in the underlying eDFM. Although scores are not defined on the basis of comparative

cubes as done in [91], they are also predefined at schema level in the eDFM (analogously to join conditions). The definition of join conditions and scores in an eDFM provides SQL based semantics. For users, it is easier to understand such a set-oriented approach based on SQL than a point-centric view as presented in [91].

The semantics of the join condition is defined as an inner join in SQL. Conceptually, comparison always concerns items of both the context of interest and the context of comparison. It is not allowed that an item of the context of interest is compared to “nothing” of the context of comparison, and vice versa. Allowing outer joins in this case would also extend the semantics of comparative analysis situations.

```

SELECT      CoI.insProvince AS CoI.insProvince,
            CoI.SumOfCosts AS CoI.SumOfCosts,
            CoI.AvgCostsPerInsurant AS CoI.AvgCostsPerInsurant,
            CoC.insProvince AS CoC.insProvince,
            CoC.SumOfCosts AS CoC.SumOfCosts,
            CoC.AvgCostsPerInsurant AS CoC.AvgCostsPerInsurant,
            CoI.SumOfCosts / CoC.SumOfCosts AS RatioOfSumOfCosts,
            CoI.AvgCostsPerInsurant / CoC.AvgCostsPerInsurant
            AS RatioOfAvgCostsPerInsurant
FROM        ( SELECT insProvince, SUM(costs) AS SumOfCosts,
                    SUM(costs) / SUM(DISTINCT insurant) AS AvgCostsPerInsurant
              FROM    DrugPrescription NATURAL JOIN Time NATURAL JOIN Insurant
              WHERE   costs / quantity > 50 AND
                    year = 2016 AND
                    inhPerSqkmInInsDistr < 400 AND
              GROUP BY insProvince
              HAVING  SUM(costs) / COUNT(DISTINCT insurant) > 1000
            ) CoI
INNER JOIN  ( SELECT insProvince, SUM(costs) AS SumOfCosts,
                    SUM(costs) / SUM(DISTINCT insurant) AS AvgCostsPerInsurant
              FROM    DrugPrescription NATURAL JOIN Time NATURAL JOIN Insurant
              WHERE   costs / quantity > 50 AND
                    year = 2015 AND
                    inhPerSqkmInInsDistr < 400 AND
              GROUP BY insProvince
            ) CoC
ON          CoI.insProvince = CoC.insProvince
WHERE      CoI.AvgCostsPerInsurant / CoC.AvgCostsPerInsurant > 1

```

Figure 3.19: Example of the translation of a comparative analysis situation into SQL

<i>CoI_insProvince</i>	<i>CoI_SumOfCosts</i>	<i>CoI_AvgCostsPerInsurant</i>	<i>CoC_insProvince</i>	<i>CoC_SumOfCosts</i>	<i>CoC_AvgCostsPerInsurant</i>	<i>RatioOfSumOfCosts</i>	<i>RatioAvgCostsPerInsurant</i>
Upper Austria	117790612.01	2120.31	Upper Austria	107790612.01	1959.21	1.0928	1.0822
Lower Austria	197790612.45	2415.07	Lower Austria	187790612.45	2295.74	1.0533	1.0520
...

Figure 3.20: Example of a result set of a comparative analysis situation

Chapter 4

Navigation Operators

Contents

4.1	Navigation Step	151
4.2	Operators Not Involving Comparison	157
4.2.1	Operators Changing Granularity Level	161
4.2.2	Operators Changing Dice Node	162
4.2.3	Operators Changing Slice Conditions	168
4.2.4	Operators Changing Base Measure Conditions	170
4.2.5	Operators Changing Aggregate Measures	173
4.2.6	Operators Changing Filter Conditions	176
4.2.7	Operator Changing Cube Access	180
4.3	Operators Involving Comparison	181
4.3.1	Operators Introducing Comparison	182
4.3.2	Operators Changing Comparison	188
4.3.3	Operators Dropping Comparison	198
4.4	Use Analysis Situations as Cubes	201
4.4.1	Derived Cubes	201
4.4.2	Enrichments of Cubes	206
4.4.3	Operator Using Non-Comparative Analysis Situations as Cubes	208

4.5 Navigation Steps Containing Navigation Guards 214

In this chapter, navigation operators for APMN4BI are introduced. Such operators describe how a business analyst navigates from an analysis situation (both non-comparative and comparative) to another one. She or he navigates from a source to a target analysis situation. A navigation operator describes the semantic difference of both.

There are navigation operators that only involve non-comparative analysis situations. For instance, OLAP operations like roll-up or drill-down give rise for simple navigation operators. Granularity level is changed to a coarser or finer one, for example, one lists measures for provinces and, in the next step, measures for districts are shown. But in combination with enrichments of an eDFM (especially, subsumption hierarchies for dimensional predicates, base measure predicates, aggregate measure predicates, and score predicates, and the sub-aggregate-measure-relation) further navigation steps can be expressed like narrowing or broadening slice conditions, or moving down to sub-measures.

Furthermore, a navigation operator can introduce, change, or lift comparisons. Such operators involve comparative analysis situations or both non-comparative and comparative ones. For instance, starting from a non-comparative analysis situation that lists drug prescription costs per province for year 2016, a business analyst can navigate to a comparative analysis situation that compares these drug prescription costs of 2016 with those of 2015.

In the sections of this chapter, we first introduce the general concept of a navigation step, i.e., the invocation of an operator for a source analysis situation which generates a target analysis situation. Afterwards, navigation operators are presented that only have non-comparative analysis situations as source and target. Navigation operators that involve both comparative and non-comparative analysis situations are described in a separate section. Subsequently, a specific operator is specified that uses non-comparative analysis situations as cubes which are based on derived cubes containing enrichments. The last section of this chapter explains navigation guards that can be used

as an additional element to control navigation steps.

In the subsequent operator specifications, **src** denotes a source analysis situation and **trg** a target analysis situation. Navigation operators can have zero, one, or more parameters depending on the operator itself. The operator definition is given by an operator name, parameters of the operator, and by pre- and postconditions over properties of source and target analysis situations that must be satisfied before and after operator invocation. A navigation operator transfers all constituents of source analysis situation **src** (which satisfies all preconditions) to target analysis situation **trg** except those items that have to be changed to satisfy the operator's postcondition. This approach of defining the semantics of navigation operators corresponds to the "frame assumption" mentioned as one option in [13] for solving the frame problem.¹

In addition to the formal definitions of navigation operators, we also give graphical representations (pictograms with operator parameters) that are used to represent a navigation step from a source to a target analysis situation graphically. Each pictogram visualizes the effect of a navigation operator.

4.1 Navigation Step

A navigation operator is defined by an operator name, formal parameters, pre- and postconditions over properties of source and target analysis situations, and the type of source and target analysis situations (non-comparative or comparative). The type of analysis situations can be considered as general pre- and postconditions. An invocation of a navigation operator represents a navigation step. The operator is applied to a source analysis situation by using actual parameters and returns a target analysis situation. Navigation guards allow additional control of navigation steps based on the result set of the source analysis situation.

¹Similar to situation calculus, the change of analysis situations via navigation operators raises the frame problem [77, 105]. Borgida et al. [13, 14] describe the frame problem and solutions with respect to procedure specifications.

This section introduces generic definitions of a navigation operator, an operator invocation, and a navigation step. Specific navigation operators are presented in the remaining sections of this chapter. First we start by a generic definition of a navigation operator.

Definition 4.1. A *navigation operator* $NAVOP = (OP, (p_1, \dots, p_q), SrcType, TrgType, PreConds, PostConds)$ comprises

1. an operator name OP ,
2. a possibly empty list of formal parameters p_1, \dots, p_q ,
3. a type $SrcType$ for the source the operator can be applied to, indicating whether this source is a non-comparative or comparative analysis situation,
4. a type $TrgType$ for the target of the operator's return value, indicating whether this target is a non-comparative or comparative analysis situation,
5. a set of preconditions $PreConds$ over properties of source analysis situations and formal parameters that have to be satisfied to obtain a valid operator application, and
6. a set of postconditions $PostConds$ over properties of target analysis situations that have to be satisfied after a valid operator application.

If $SrcType$ and $TrgType$ indicate a non-comparative analysis situation for both source and target, navigation operator $NAVOP$ is also called *non-comparative navigation operator*.² In the other cases (where either $SrcType$ or $TrgType$, or both indicate a comparative analysis situation), navigation operator $NAVOP$ is also called *comparative navigation operator*.³

²The specific non-comparative navigation operators are defined in Section 4.2 and Subsection 4.4.3 where also the types for the formal parameters p_1, \dots, p_q , and the operators' preconditions $PreConds$ and postconditions $PostConds$ are specified.

³The specific comparative navigation operators are defined in Section 4.3 where also the types for the formal parameters p_1, \dots, p_q , and the operators' preconditions $PreConds$ and postconditions $PostConds$ are specified.

Specific comparative navigation operators of Subsections 4.3.1 and 4.3.2, additionally require that the first formal parameter p_1 of the operator definition *NAVOP* represents a non-comparative navigation operator.

As presented in the operator specifications of the subsequent Sections 4.2, 4.3, and 4.4, the operator name OP and the list of formal parameters p_1, \dots, p_q uniquely define a navigation operator (also including source type, target type, preconditions, and postconditions); thus, it is also allowed to write $OP(p_1, \dots, p_q)$ for navigation operator $(OP, (p_1, \dots, p_q), SrcType, TrgType, PreConds, PostConds)$. Moreover, we define $SrcType(OP(p_1, \dots, p_q)) = SrcType$, $TrgType(OP(p_1, \dots, p_q)) = TrgType$, $PreConds(OP(p_1, \dots, p_q)) = PreConds$, and $PostConds(OP(p_1, \dots, p_q)) = PostConds$. \square

The following generic definition introduces the application of a navigation operator. A valid operator invocation takes actual parameters with respect to the list of formal parameters and satisfies all preconditions. The operator is applied to a source analysis situation and returns a target analysis situation that fulfills all postconditions.

Definition 4.2. Let $NAVOP = (OP, (p_1, \dots, p_q), SrcType, TrgType, PreConds, PostConds)$ be a navigation operator. Expression $\mathbf{src}.OP(\bar{p}_1, \dots, \bar{p}_q)$ defines an *invocation* of navigation operator *NAVOP* that takes actual parameters $\bar{p}_1, \dots, \bar{p}_q$ with respect to the operator's list of formal parameters p_1, \dots, p_q and that is applied to an analysis situation \mathbf{src} of source type $SrcType$. The invocation returns an analysis situation of target type $TrgType$ that satisfies all postconditions in $PostConds$.

If *NAVOP* represents a comparative navigation operator and if formal parameter p_1 represents a non-comparative navigation operator $OP'(p'_1, \dots, p'_q)$, it is additionally required that actual parameter \bar{p}_1 represents an expression $OP'(\bar{p}'_1, \dots, \bar{p}'_q)$ that specifies an invocation of non-comparative navigation operator p_1 applied to $CoI_{\mathbf{src}}$ or $CoC_{\mathbf{src}}$ (or to both) and with actual parameters $\bar{p}'_1, \dots, \bar{p}'_q$. In this case, we also consider expression $OP'(\bar{p}'_1, \dots, \bar{p}'_q)$ as an invocation of operator $OP'(p'_1, \dots, p'_q)$ —implicitly applied to $CoI_{\mathbf{src}}$ or $CoC_{\mathbf{src}}$ (or to both).

The invocation $\mathbf{src}.OP(\bar{p}_1, \dots, \bar{p}_q)$ is *valid*, if all preconditions in $PreConds$

are fulfilled. In the case that $NAVOP$ represents a comparative navigation operator and if formal parameter p_1 represents a non-comparative navigation operator, it is additionally required that actual parameter \bar{p}_1 also represents a valid invocation of non-comparative navigation operator p_1 . \square

A navigation step comprises an operator invocation, the source analysis situation the operator is applied to, the target analysis situation it returns, and a boolean expression denoted as navigation guard. This is formalized in the following generic definition.

Definition 4.3. Let $NAVOP = (OP, (p_1, \dots, p_q), SrcType, TrgType, PreConds, PostConds)$ be a navigation operator. A navigation step $\mathbf{nav} = (\mathbf{src}.OP(\bar{p}_1, \dots, \bar{p}_q), \mathbf{trg}, NavGrd)$ comprises

1. a valid invocation $\mathbf{src}.OP(\bar{p}_1, \dots, \bar{p}_q)$ of navigation operator $NAVOP$,
2. a target analysis situation \mathbf{trg} such that $\mathbf{trg} = \mathbf{src}.OP(\bar{p}_1, \dots, \bar{p}_q)$, and
3. a boolean expression $NavGrd$ (*navigation guard*) defined over \mathbf{src} using operators defined in Table 4.1 that evaluate (in its entirety) result set $ResultSet_{\mathbf{src}}$ of source analysis situation \mathbf{src} (boolean expression \mathbf{true} allows that the navigation guard is always true, i.e., that it is always satisfied).

We provide the following alternative syntax of an operator invocation to include the notion of navigation guards: $\mathbf{src}.[NavGrd] OP(\bar{p}_1, \dots, \bar{p}_q)$. Furthermore, in the context of navigation step \mathbf{nav} , we define $Source_{\mathbf{nav}} = \mathbf{src}$, $Target_{\mathbf{nav}} = \mathbf{trg}$, and $NavGrd_{\mathbf{nav}} = NavGrd$. \square

The generic definitions of an operator invocation and of a navigation step uses an object-oriented notation. Operator OP takes values $\bar{p}_1, \dots, \bar{p}_q$ as actual parameters and is applied to source analysis situation \mathbf{src} . The target analysis situation is represented by operator call $\mathbf{src}.OP(\bar{p}_1, \dots, \bar{p}_q)$ and is explicitly indicated in the navigation step \mathbf{nav} as the second component \mathbf{trg} .

The application of a navigation operator to an analysis situation generates another analysis situation to which again a navigation operator can be

applied. Thus, it is also allowed to write sequences of operator invocations in an object-oriented style: $\mathbf{src}.OP_1(\bar{p}_1^1, \dots, \bar{p}_{q_1}^1) \dots OP_n(\bar{p}_1^n, \dots, \bar{p}_{q_n}^n)$. This sequence of operator invocation corresponds to a sequence of navigation steps $(\mathbf{src}.OP_1(\bar{p}_1^1, \dots, \bar{p}_{q_1}^1), \mathbf{trg}_1, NavGrd_1), \dots, (\mathbf{trg}_{n-1}.OP_n(\bar{p}_1^n, \dots, \bar{p}_{q_n}^n), \mathbf{trg}_n, NavGrd_n)$ where the target of one navigation step becomes the source of the next one.⁴

The notion of a navigation guard allows to control the application (invocation) of a navigation step. It represents a necessary condition for executing the target analysis situation of a navigation step.⁵ If boolean expression $NavGrd$ is evaluated to true, query $Query_{\mathbf{trg}}$ of target analysis situation \mathbf{trg} can be executed, otherwise $Query_{\mathbf{trg}}$ must not be executed. This evaluation depends on result set $ResultSet_{\mathbf{src}}$ of source analysis situation \mathbf{src} . One can also think of controlling the operator invocation meaning that operator OP need not be invoked, if the evaluation of the navigation guard $NavGrd$ yields false, and, in this case, also no target analysis situation need to be created because no query must be executed. As we allow to insert the navigation guard in the notation of an operator invocation, a sequence of operator invocations as presented above can be written as $\mathbf{src}.[NavGrd_1]OP_1(\bar{p}_1^1, \dots, \bar{p}_{q_1}^1) \dots [NavGrd_n]OP_n(\bar{p}_1^n, \dots, \bar{p}_{q_n}^n)$. If the navigation guard is equal to boolean expression \mathbf{true} , we usually omit the syntactical part $[NavGrd]$, i.e., instead of $\mathbf{src}.[\mathbf{true}]OP(\bar{p}_1, \dots, \bar{p}_q)$, one again simply writes $\mathbf{src}.OP(\bar{p}_1, \dots, \bar{p}_q)$.

Although, in this chapter, we do not yet have formal semantics about execution of a sequence of analysis situations connected by navigation steps, the notion of navigation guards can be interpreted in the following way: A navigation guard examines the result set of a source analysis situation. If the navigation guard is evaluated to false, the navigation step is not invoked, the query of the target analysis situation of the navigation step cannot be exe-

⁴Note that we do not define sequences of operator invocations but we provide a simplification for writing sequences of navigation steps that can be interrupted by navigation guards.

⁵Note, a navigation guard represents a necessary but not a sufficient condition for execution of the target analysis situation. Pre- and postconditions of a navigation step have to be fulfilled, too.

Table 4.1: Operators used in navigation guard expressions of a non-comparative analysis situation **as** or comparative analysis situation **cas**. We use an object-oriented style to indicate the application of such an operator to **as** and **cas**.

Operator	Definition
<code>as.hasResult()</code>	$ResultSet_{\mathbf{as}} \neq \emptyset$
<code>cas.hasResult()</code>	$ResultSet_{\mathbf{cas}} \neq \emptyset$
<code>as.hasNoResult()</code>	$ResultSet_{\mathbf{as}} = \emptyset$
<code>cas.hasNoResult()</code>	$ResultSet_{\mathbf{cas}} = \emptyset$

cuted, and the navigation is interrupted meaning that all other subsequent navigation steps are not invoked. In this sense, navigation guards additionally control navigation. At this point, note that also the evaluation of pre- and postconditions of a navigation step are necessary conditions for application (invocation), i.e., a navigation guard represents a necessary but not a sufficient condition for invoking a navigation step.

In the graphical notation, a navigation step is depicted by graphical representations of the source and target analysis situation introduced in Chapter 3, and by an arrow from the source to the target including the labelled operator symbol and an actual parameter description. A navigation guard (except navigation guards containing the simple boolean expression `true`) is depicted by a rectangle (containing a grey diamond in the left upper corner) that precedes the operator symbol. Figure 4.1 gives several examples of the graphical representation of navigation steps from non-comparative to other non-comparative analysis situations. Graphical representations of examples comprising comparative navigation steps and comprising specific operator `useAsCube` are demonstrated in Sections 4.3 and 4.4. An example of a navigation step containing a navigation guard is presented in Figure 4.12 (see Section 4.5).

After presenting a generic definition of navigation steps, specific navigation operators and specific navigation steps are specified and demonstrated. In the two subsequent sections, we introduce navigation operators not involving comparison (Section 4.2) and navigation operators used to define navigation steps comprising comparative analysis situations (Section 4.3). Section

4.4 presents specific operator `useAsCube` that takes a non-comparative analysis situation as a cube in the target analysis situation of a navigation step. Finally, in Section 4.5, navigation steps containing navigation guards are explained.

4.2 Operators Not Involving Comparison

In this section, we present seven groups of navigation operators that only involve non-comparative analysis situations: navigation operators changing granularity, navigation operators changing dice node, navigation operators changing slice conditions, navigation operators changing filter conditions for base measures, navigation operators changing measures, navigation operators changing filter conditions, and one navigation operator changing cube access. Figure 4.1 shows examples of navigation steps comprising operators of each operator group.

For all operators presented in this section, we require that both `src` and `trg` are non-comparative analysis situations. Furthermore, for operator parameter D , we suppose precondition $D \in DimSchemas_{src}$. Tables 4.2 – 4.10 contain the formal definition and the graphical representation of each navigation operator.

The formal definition comprises the operator name and parameters required for operator invocation⁶, the precondition that must be fulfilled for source analysis situation `src` and operation parameters, and the postcondition that is satisfied for target analysis situation `trg` after operator execution. Remember, as mentioned in the introduction of this chapter, we postulate the frame assumption such that nothing else of target analysis situation `trg` differs from the source analysis situation `src` except those things claimed by the postcondition.

The operator name expresses the action that has to be done to transform source analysis situation `src` into target analysis situation `trg` using additional information given by parameters. Operator names are formed by

⁶The operator invocation (navigation step) is underlined in this table column.

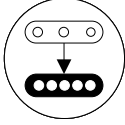
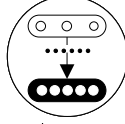

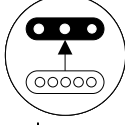
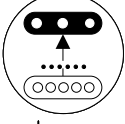

letters and a small set of specific symbols (instead of words) that express specific intentions: symbol “+” expresses to add something, symbol “-” expresses to remove something, and symbol “->” expresses to change something from an old to a new value. The position of a parameter in an operator invocation determines the type of the parameter. Some operators have a set of values (of a certain type) as a parameter. Each parameter refers to a constituent of a non-comparative analysis situation (cube, base measure condition, aggregate measure, filter condition, dimension, dice level, dice node, dice level, slice condition, and granularity level).

In the second column of Tables 4.2 – 4.10, one can find the graphical representation of navigation operators. The whole information of an operator invocation as given in the definition column (by operator name and parameters) is also depicted in the graphical representation. Additionally, this representation contains a pictogram that expresses graphically the meaning of the operator’s action. The pictogram can be considered as a graphical synonym of the operator name. Within these pictograms, symbols for constituents of analysis situations are contained twice with different filling (unfilled and black-filled) and decorated by other symbols (for instance, by arrows). These symbolizations express the change of constituents of non-comparative analysis situations.

Operator parameters are given below the pictogram and they are graphically prefixed by symbols as used for the constituents of non-comparative analysis situations. Thus, instead of the parameter position in the operator invocation (as given in the definition column), in the graphically representation, the graphical prefix determines the parameter type. As for constituents in the lean graphical notation of analysis situations, also the parameters of the graphical representation of navigation operations can be graphically positioned in a loose and flexible manner. Operators (with symbol “->” in their name) that exchange an old value for a new value (given as two parameters) require a specific graphical notation to express both the old and the new parameter value. In this cases the parameter representing the old value is linked by symbol “->” with the parameter representing the new value.

In the following subsections, we continue to describe each navigation op-

Table 4.2: Operators `drillDownOneLevel`, `drillDownToLevel`, `rollUpOneLevel`, and `rollUpToLevel`

Operator Definition	Symbol
<u><code>drillDownOneLevel(D)</code></u> Precondition: $GranLvl_{src}(D) \neq base_D$ Postcondition: $GranLvl_{trg}(D) \rightarrow GranLvl_{src}(D)$	<code>drillDownOneLevel</code>  $\swarrow D$
<u><code>drillDownToLevel(D, G)</code></u> Precondition: $G \rightarrow GranLvl_{src}(D)$ Postcondition: $GranLvl_{trg}(D) = G$	<code>drillDownToLevel</code>  $\swarrow D$  G
<u><code>rollUpOneLevel(D)</code></u> Precondition: $GranLvl_{src}(D) \neq top_D$ Postcondition: $GranLvl_{src}(D) \rightarrow GranLvl_{trg}(D)$	<code>rollUpOneLevel</code>  $\swarrow D$
<u><code>rollUpToLevel(D, G)</code></u> Precondition: $GranLvl_{src}(D) \rightarrow G$ Postcondition: $GranLvl_{trg}(D) = G$	<code>rollUpToLevel</code>  $\swarrow D$  G

erator including the formal definitions and graphical representations. The subsections are organized with respect to the operator groups: operators changing granularity level, operators changing dice node, operators changing slice conditions, operators changing base measure conditions, operators changing aggregate measures, operators changing filter conditions, and one operator for changing the cube access.

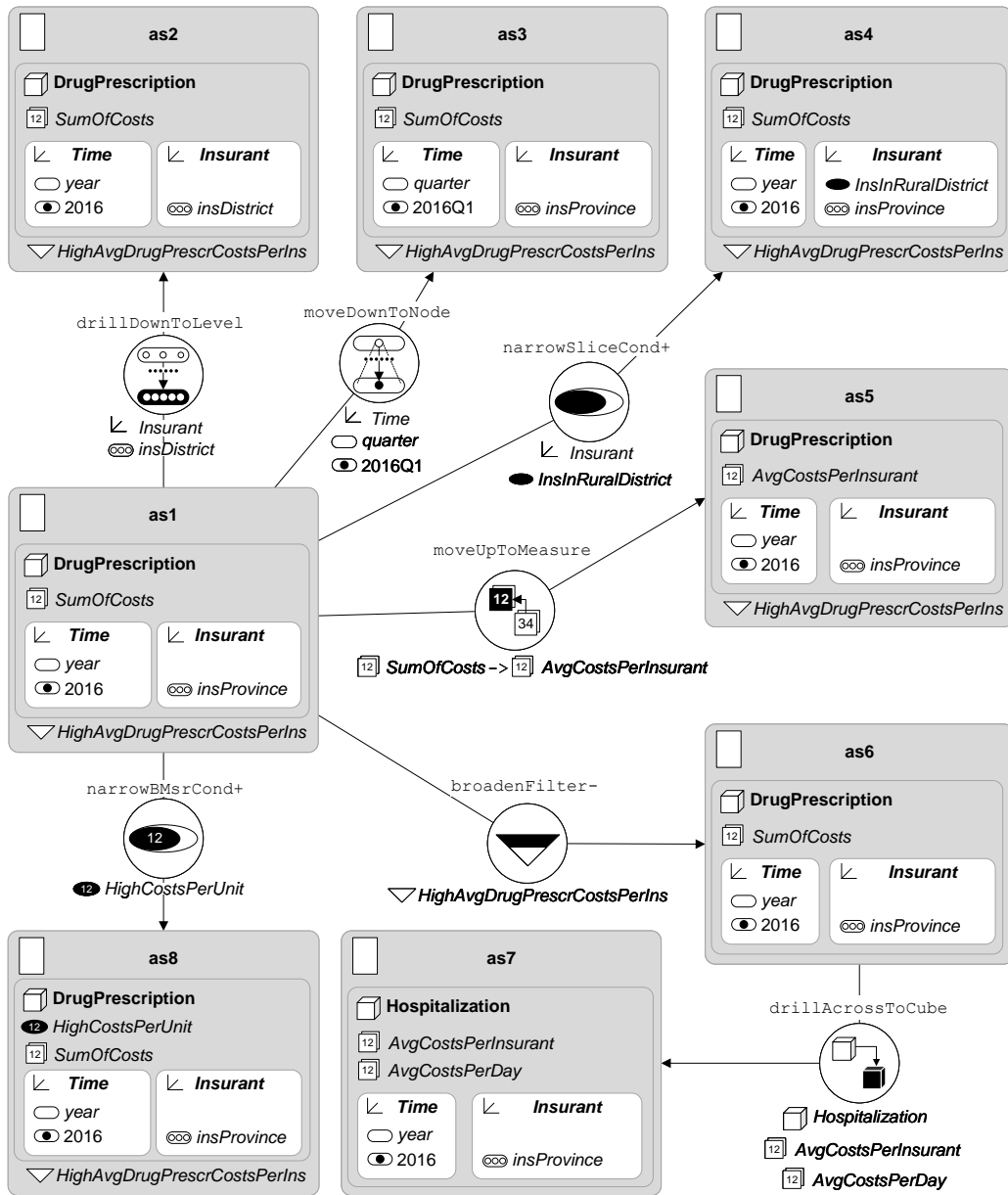


Figure 4.1: Navigation operator examples

4.2.1 Operators Changing Granularity Level

The granularity level of a non-comparative analysis situation can be changed by drill-down and roll-up operators. Drill-down refines a granularity level of a dimension whereas roll-up changes to a coarser granularity level. Operator `drillDownOneLevel` changes the granularity level of a dimension to the next finer one and `drillDownToLevel` refines to an arbitrary level of a dimension. The corresponding inverse operators are `rollUpOneLevel` and `rollUpToLevel`.

The definition of those operators can be found in Table 4.2. As preconditions, the source analysis situation must not have a base granularity level in the case of drill-down operations and it must not have a top level in the case of roll-up operations—in both cases with respect to dimension schema D . For operators `drillDownToLevel` and `rollUpToLevel`, the preconditions require appropriate roll-up relationships.

In the example of Figure 4.1, there is a `drillDownToLevel`-operator from source analysis situation `as1` to target analysis situation `as2` with dimension parameter *Insurant* and parameter *insDistrict* as granularity level. The operator transfers all information of `as1` to `as2` except granularity level in dimension *Insurant* which is changed from *insProvince* to *insDistrict*. The operator invocation is valid because the precondition that level *insDistrict* rolls up to *insProvince* is satisfied. Formally, this navigation step can be written as `as1.drillDownToLevel(Insurant, insDistrict)`. This navigation step returns target analysis situation `as2`. Thus, we can write: `as2 = as1.drillDownToLevel(Insurant, insDistrict)`.

In the example of Figure 4.1, instead of `drillDownToLevel` one could also use operator `drillDownOneLevel` where granularity level parameter is omitted. In this case, one can write `as2 = as1.drillDownOneLevel(Insurant)`. The roll-up operators can be considered as inverse operators of drilling down. Thus, for obtaining non-comparative analysis situation `as1` from `as2`, one can model navigation step `as1 = as2.rollUpToLevel(Insurant, insProvince)` or, alternatively, `as1 = as2.rollUpOneLevel(Insurant)`.

4.2.2 Operators Changing Dice Node

The navigation operators of Table 4.3, 4.4, and 4.5 change the dice node and maybe the dice level of a non-comparative analysis situation. Operators `moveDownToNode` and `moveUpToNode` move to an arbitrary subnode or to the unique supernode, respectively. In both cases, dice node and dice level are changed (with respect to the sub-super-node-relation of a dimension instance). Dimension, dice level, and (in the case of operator `moveDownToNode`) dice node are given as parameters. Navigation operator `moveAsideToNode` does not change the dice level but only the dice node of a dimension. As a constraint, the dice node of the source and the target must have the same direct supernode. A more general navigation operator that navigates to an arbitrary node within a dimension is operator `moveToNode` with parameters for dimension, dice level, and dice node. This operator has no further constraints.

Operators `moveDownToFirstNode`, `moveDownToLastNode`, `moveToNextNode`, and `moveToPrevNode` (see Table 4.4 and 4.5) are based on the order relation \prec defined on the nodes of the dice level. In the postconditions of these operators, we implicitly introduced minimum, maximum, next, and previous operators with respect to order relation \prec : min_{\prec} , max_{\prec} , $next_{\prec}$, and $prev_{\prec}$. Each of the navigation operators of Table 4.4 and 4.5 has an overloaded version with a dimension level as an additional parameter. Operators `moveDownToFirstNode` and `moveDownToLastNode` move down to the first or to the last direct subnode, or to the first or to the last subnode of level L (in the case of the overloaded operator version with dimension level L as an additional parameter). Navigation operators `moveToNextNode` and `moveToPrevNode` move to the next or previous node with respect to the common supernode. This common supernode represents the direct supernode or the supernode of level L (in the case of the overloaded version with dimension level L as additional parameter).

In Figure 4.1, one finds the navigation operation `moveDownToNode` from analysis situation `as1` to `as3` with dimension *Time*, dice level *quarter*, and dice node `2016Q1` as parameters. The operator changes in the dimension

Table 4.3: Operators `moveDownToNode`, `moveUpToNode`, `moveAsideToNode`, and `moveToNode`

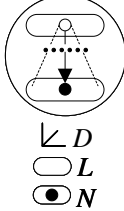
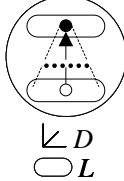
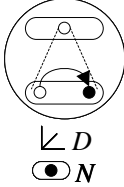
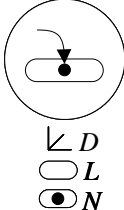
Operator Definition	Symbol
<p><u><code>moveDownToNode(D, L, N)</code></u> Precondition: $L \in Lvl_D$, $L \rightarrow DiceLvl_{src}(D)$, $N \in NodesOfLvl_d(L)$ with $d = DimInstance_{src}(D)$, and $N \rightarrow DiceNode_{src}(D)$. Postcondition: $L = DiceLvl_{trg}(D)$ and $N = DiceNode_{trg}(D)$.</p>	<p><code>moveDownToNode</code></p> 
<p><u><code>moveUpToNode(D, L)</code></u> Precondition: $L \in Lvl_D$, $L' \rightarrow L$, and $N' \rightarrow N$, where $L' = DiceLvl_{src}(D)$, $N' = DiceNode_{src}(D)$, and $N = SuperNodeOf_{L',L}^d(N')$ with $d = DimInstance_{src}(D)$. Postcondition: $L = DiceLvl_{trg}(D)$ and $N = DiceNode_{trg}(D)$.</p>	<p><code>moveUpToNode</code></p> 
<p><u><code>moveAsideToNode(D, N)</code></u> Precondition: $L' \neq top_D$, where $L' = DiceLvl_{src}(D)$, $N \in NodesOfLvl_d(L')$ with $d = DimInstance_{src}(D)$ and $L' \rightarrow L$; $N \neq DiceNode_{src}(D)$ and $SuperNodeOf_{L',L}^d(N) =$ $SuperNodeOf_{L',L}^d(DiceNode_{src}(D))$. Postcondition: $DiceNode_{trg}(D) = N$.</p>	<p><code>moveAsideToNode</code></p> 
<p><u><code>moveToNode(D, L, N)</code></u> Precondition: $N \neq DiceNode_{src}(D)$, $L \in Lvl_D$, and $N \in$ $Nodes_d(L)$ with $d = DimInstance_{src}(D)$. Postcondition: $DiceLvl_{trg}(D) = L$ and $DiceNode_{trg}(D) = N$.</p>	<p><code>moveToNode</code></p> 

Table 4.4: Operators `moveDownToFirstNode` and `moveDownToLastNode`

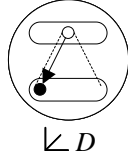
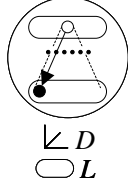
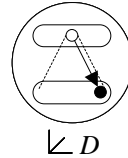
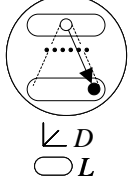
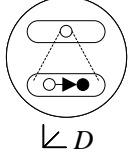
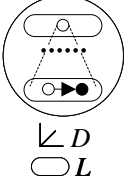
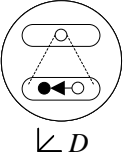
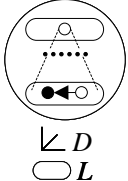
Operator Definition	Symbol
<u><code>moveDownToFirstNode(D)</code></u> Precondition: $DiceLvl_{src}(D) \neq base_D$ Postcondition: $DiceLvl_{trg}(D) \rightarrow DiceLvl_{src}(D)$ and $DiceNode_{trg}(D) =$ $min_{\prec} \{N \mid N \rightarrow DiceNode_{src}(D)\}.$	<code>moveDownToFirstNode</code>  $\swarrow D$
<u><code>moveDownToFirstNode(D, L)</code></u> Precondition: $L \in Lvl_D$ and $L \rightarrow DiceLvl_{src}(D).$ Postcondition: $DiceLvl_{trg}(D) = L$ and $DiceNode_{trg}(D) =$ $min_{\prec} \{N \mid N \in NodesOfLvl_d(L) \text{ with}$ $\mathbf{d} = DimInstance_{src}(D),$ $N \rightarrow DiceNode_{src}(D)\}.$	<code>moveDownToFirstNode</code>  $\swarrow D$ $\bigcirc L$
<u><code>moveDownToLastNode(D)</code></u> Precondition: $DiceLvl_{src}(D) \neq base_D$ Postcondition: $DiceLvl_{trg}(D) \rightarrow DiceLvl_{src}(D)$ and $DiceNode_{trg}(D) =$ $max_{\prec} \{N \mid N \rightarrow DiceNode_{src}(D)\}.$	<code>moveDownToLastNode</code>  $\swarrow D$
<u><code>moveDownToLastNode(D, L)</code></u> Precondition: $L \in Lvl_D$ and $L \rightarrow DiceLvl_{src}(D).$ Postcondition: $DiceLvl_{trg}(D) = L$ and $DiceNode_{trg}(D) =$ $max_{\prec} \{N \mid N \in NodesOfLvl_d(L) \text{ with}$ $\mathbf{d} = DimInstance_{src}(D),$ $N \rightarrow DiceNode_{src}(D)\}.$	<code>moveDownToLastNode</code>  $\swarrow D$ $\bigcirc L$

Table 4.5: Operators `moveToNextNode` and `moveToPrevNode`

Operator Definition	Symbol
<p><u><code>moveToNextNode(D)</code></u> Precondition: Let $L' = \text{DiceLvl}_{\text{src}}(D) \neq \text{top}_D$, $N = \text{SuperNodeOf}_{L',L}^{\mathbf{d}}(\text{DiceNode}_{\text{src}}(D))$ with $\mathbf{d} = \text{DimInstance}_{\text{src}}(D)$ and $L' \rightarrow L$, and $N' \in \text{NodesOfLvl}_{\mathbf{d}}(L')$ with $N' \rightarrow N$ and $N' = \text{next}_{\prec}(\text{DiceNode}_{\text{src}}(D))$. Postcondition: $\text{DiceNode}_{\text{trg}}(D) = N'$</p>	<p><code>moveToNextNode</code></p> 
<p><u><code>moveToNextNode(D, L)</code></u> Precondition: $L \in \text{Lvls}_D$; let $L' = \text{DiceLvl}_{\text{src}}(D) \neq \text{top}_D$, $N = \text{SuperNodeOf}_{L',L}^{\mathbf{d}}(\text{DiceNode}_{\text{src}}(D))$ with $\mathbf{d} = \text{DimInstance}_{\text{src}}(D)$ and $L' \rightarrow L$, and $N' \in \text{NodesOfLvl}_{\mathbf{d}}(L')$ with $N' \rightarrow N$ and $N' = \text{next}_{\prec}(\text{DiceNode}_{\text{src}}(D))$. Postcondition: $\text{DiceNode}_{\text{trg}}(D) = N'$</p>	<p><code>moveToNextNode</code></p> 
<p><u><code>moveToPrevNode(D)</code></u> Precondition: Let $L' = \text{DiceLvl}_{\text{src}}(D) \neq \text{top}_D$, $N = \text{SuperNodeOf}_{L',L}^{\mathbf{d}}(\text{DiceNode}_{\text{src}}(D))$ with $\mathbf{d} = \text{DimInstance}_{\text{src}}(D)$ and $L' \rightarrow L$, and $N' \in \text{NodesOfLvl}_{\mathbf{d}}(L')$ with $N' \rightarrow N$ and $N' = \text{prev}_{\prec}(\text{DiceNode}_{\text{src}}(D))$. Postcondition: $\text{DiceNode}_{\text{trg}}(D) = N'$</p>	<p><code>moveToPrevNode</code></p> 
<p><u><code>moveToPrevNode(D, L)</code></u> Precondition: $L \in \text{Lvls}_D$; let $L' = \text{DiceLvl}_{\text{src}}(D) \neq \text{top}_D$, $N = \text{SuperNodeOf}_{L',L}^{\mathbf{d}}(\text{DiceNode}_{\text{src}}(D))$ with $\mathbf{d} = \text{DimInstance}_{\text{src}}(D)$ and $L' \rightarrow L$, and $N' \in \text{NodesOfLvl}_{\mathbf{d}}(L')$ with $N' \rightarrow N$ and $N' = \text{prev}_{\prec}(\text{DiceNode}_{\text{src}}(D))$. Postcondition: $\text{DiceNode}_{\text{trg}}(D) = N'$</p>	<p><code>moveToPrevNode</code></p> 

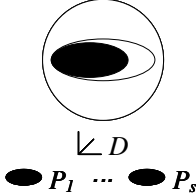
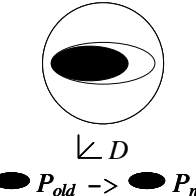
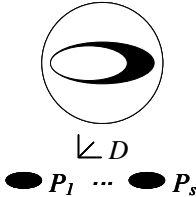
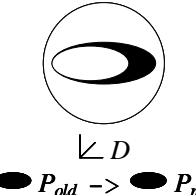
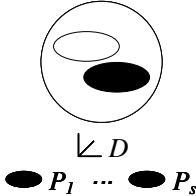
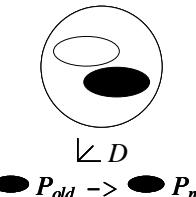
qualification of dimension *Time* dice level and dice node from year 2016 to the first quarter of 2016. This navigation step can be written as `as1.moveDownToNode(Time, quarter, 2016Q1)` and it returns target analysis situation `as3`, such that we can also write `as3 = as1.moveDownToNode(Time, quarter, 2016Q1)`. We assume to have a temporal order relation on the quarters of a year. Thus, one could also apply operator `moveDownToFirstNode` to navigate to the first quarter of year 2016.

We demonstrate further examples (in the formal notation) which are not depicted in Figure 4.1. Non-comparative analysis situation `as1` can be obtained from `as3` by executing navigation step `as3.moveUpToNode(Time, year)` that can be considered as an inverse operation with respect to the move-down operation shown in Figure 4.1. In this case, the dice node in dimension *Time* is changed to year 2016 because dice node 2016Q1 belongs to super node 2016.

Starting from `as3` as source analysis situation, one can change to the third quarter of year 2016 by applying operator `moveAsideToNode` in the following way: `as3.moveAsideToNode(Time, 2016Q3)`. This can be done because nodes 2016Q1 (source) and 2016Q3 (target) belong the same super node 2016. An arbitrary navigation to, for instance, dice node *May 2015* can be attained by application of operator `moveToNode`: `as3.moveToNode(Time, May 2015)`.

If one wants to navigate through all quarters of year 2016, she or he can execute the following operation sequence: `as1.moveDownToFirstNode(Time).moveToNextNode(Time).moveToNextNode(Time).moveToNextNode(Time)`. With respect to the graphical notation, such sequences of navigation steps have to be depicted as a sequence of five analysis situations (starting with analysis situation `as1`) connected by the appropriate navigation operation. In the next chapter, we show how such sequences can be drawn in a compact way at schema level.

Table 4.6: Operators narrowSliceCond+ , narrowSliceCond- , broadenSliceCond- , broadenSliceCond- , refocusSliceCond , and refocusSliceCond-

Operator Definition	Symbol
$\text{narrowSliceCond+}(D, \{P_1, \dots, P_s\})$ Precondition: $P_1, \dots, P_s \in \text{DimPredicates}_D$ and $\{P_1, \dots, P_s\} \cap \text{SliceConds}_{\text{src}}(D) = \emptyset$. Postcondition: $\text{SliceConds}_{\text{trg}}(D) =$ $\text{SliceConds}_{\text{src}}(D) \cup \{P_1, \dots, P_s\}$	narrowSliceCond+ 
$\text{narrowSliceCond-}(D, P_{\text{old}}, P_{\text{new}})$ Precondition: $P_{\text{old}}, P_{\text{new}} \in \text{DimPredicates}_D$, $P_{\text{old}} \in \text{SliceConds}_{\text{src}}(D)$, $P_{\text{new}} \notin \text{SliceConds}_{\text{src}}(D)$, and $P_{\text{new}} \Rightarrow P_{\text{old}}$. Postcondition: $\text{SliceConds}_{\text{trg}}(D) =$ $\text{SliceConds}_{\text{src}}(D) \setminus \{P_{\text{old}}\} \cup \{P_{\text{new}}\}$	narrowSliceCond- 
$\text{broadenSliceCond-}(D, \{P_1, \dots, P_s\})$ Precondition: $P_1, \dots, P_s \in \text{DimPredicates}_D$ and $\{P_1, \dots, P_s\} \subseteq \text{SliceConds}_{\text{src}}(D)$. Postcondition: $\text{SliceConds}_{\text{trg}}(D) =$ $\text{SliceConds}_{\text{src}}(D) \setminus \{P_1, \dots, P_s\}$	broadenSliceCond- 
$\text{broadenSliceCond-}(D, P_{\text{old}}, P_{\text{new}})$ Precondition: $P_{\text{old}}, P_{\text{new}} \in \text{DimPredicates}_D$, $P_{\text{old}} \in \text{SliceConds}_{\text{src}}(D)$, $P_{\text{new}} \notin \text{SliceConds}_{\text{src}}(D)$, and $P_{\text{old}} \Rightarrow P_{\text{new}}$. Postcondition: $\text{SliceConds}_{\text{trg}}(D) =$ $\text{SliceConds}_{\text{src}}(D) \setminus \{P_{\text{old}}\} \cup \{P_{\text{new}}\}$	broadenSliceCond- 
$\text{refocusSliceCond}(D, \{P_1, \dots, P_s\})$ Precondition: $P_1, \dots, P_s \in \text{DimPredicates}_D$ and $\{P_1, \dots, P_s\} \neq \text{SliceConds}_{\text{src}}(D)$. Postcondition: $\text{SliceConds}_{\text{trg}}(D) =$ $\{P_1, \dots, P_s\}$	refocusSliceCond 
$\text{refocusSliceCond-}(D, P_{\text{old}}, P_{\text{new}})$ Precondition: $P_{\text{old}}, P_{\text{new}} \in \text{DimPredicates}_D$, $P_{\text{old}} \in \text{SliceConds}_{\text{src}}(D)$, and $P_{\text{new}} \notin \text{SliceConds}_{\text{src}}(D)$. Postcondition: $\text{SliceConds}_{\text{trg}}(D) =$ $\text{SliceConds}_{\text{src}}(D) \setminus \{P_{\text{old}}\} \cup \{P_{\text{new}}\}$	refocusSliceCond- 

4.2.3 Operators Changing Slice Conditions

Slice conditions can be narrowed, broadened, or arbitrarily refocused. Table 4.6 lists six operators used for changing the set of slice conditions in the dimension qualification: `narrowSliceCond+`, `narrowSliceCond->`, `broadenSliceCond-`, `broadenSliceCond->`, `refocusSliceCond`, and `refocusSliceCond->`. In the case of `narrowSliceCond+` and `narrowSliceCond->`, the overall slice condition of the target implies the overall slice condition of the source, whereas, if operator `broadenSliceCond-` or `broadenSliceCond->` is applied, the overall slice condition of the source implies the one of the target. Navigation operators `refocusSliceCond` and `refocusSliceCond->` have no constraints with respect to such implications. Note that dimensional predicates of the eDFM are used as slice conditions and implications are derived from the predicate hierarchy.

The operators of this group have dimension and dimensional predicates as parameters. Operator `narrowSliceCond+` receives a set of dimensional predicates P_1, \dots, P_s that are added to the slice conditions of the source analysis situation. As a result, the overall slice condition of the target analysis situation implies the overall slice condition of the source analysis situation or, in other words, the overall slice condition of the source analysis situation subsumes the overall slice condition of the target analysis situation. In the second version of narrowing by operator `narrowSliceCond->`, only one dimensional predicate P_{old} of the source analysis situation is exchanged by another dimensional predicate P_{new} . Because the precondition requires that P_{new} implies P_{old} , the overall slice condition of the source analysis situation subsumes the overall slice condition of the target analysis situation.

In a similar way, one can broaden the overall slice condition of a source analysis situation. Navigation operator `broadenSliceCond-` receives a set of dimensional predicates P_1, \dots, P_s that are part of the slice conditions of the source analysis situation and that are removed from it. As a consequence, the overall slice condition of the target analysis situation subsumes the overall slice condition of the source analysis situation. Operator `broadenSliceCond->` exchanges a dimensional predicate P_{old} of the source analysis situa-

tion by a dimensional predicate P_{new} that is implied by P_{old} .

The last type of changing slice conditions of a source analysis situation is to make arbitrary changes regardless of implications between the slice conditions of the source and target analysis situation. Operator **refocusSliceCond** removes all slice conditions of the source analysis situation and adds new dimensional predicates P_1, \dots, P_s given as a set valued parameter. In the second version of refocusing (operator **refocusSliceCond->**), a single slice condition P_{old} in the slice condition set of the source analysis situation is exchanged by another dimensional predicate P_{new} —again regardless of possible implications.

In our example of Figure 4.1, operator **narrowSliceCond+** is applied to analysis situation **as1** with dimension parameter *Insurant* and dimensional predicate parameter *InsInRuralDistrict*. Thus, in target analysis situation **as4**, slice condition *InsInRuralDistrict* is added. The target analysis situation is restricted to insurants living in rural districts. The slice conditions of target analysis situation **as4** implies the empty set of slice conditions of source analysis situation **as1**, i.e., *InsInRuralDistrict* implies *true*. Formally, this navigation step is expressed by **as1.narrowSliceCond+(Insurant, {InsInRuralDistrict})**. In this example, the set of dimensional predicates in the second parameter only comprises one dimensional predicate.

If there is a requirement to narrow the slice condition of dimension schema *Insurant* in analysis situation **as1** to old insurants in rural districts, a set with two dimensional predicates *InsInRuralDistrict* and *OldInsurant* can be used as parameter: **trg** $\stackrel{def}{=} \mathbf{as1.narrowSliceCond+(Insurant, \{InsInRuralDistrict, OldInsurant\})}$. Analysis situation **trg** can be broadened by removing dimensional predicate *OldInsurant* in dimension schema *Insurant* such that one obtains an analysis situation equal to analysis situation **as4**, i.e., **as4 = trg.broadenSliceCond-(Insurant, {InsInRuralDistrict})**.

The slice condition of dimension schema *Insurant* in analysis situation **as1** can also be narrowed to old insurants in rural districts by using dimensional predicate *OldInsInRuralDistrict*: **trg** $\stackrel{def}{=} \mathbf{as1.narrowSliceCond+(Insurant, \{OldInsInRuralDistrict\})}$. If one intends to broaden target analysis situation **trg** to all insurants in rural districts, navigation operator

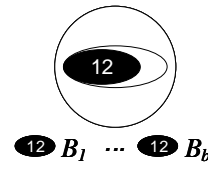
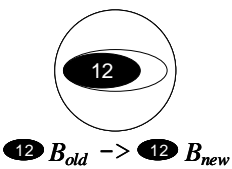
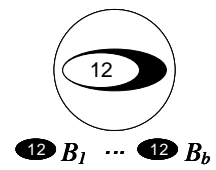
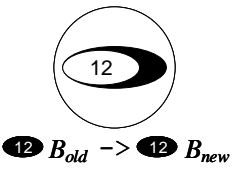
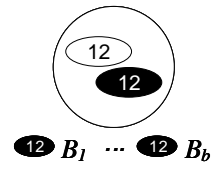
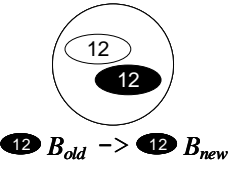
`broadenSliceCond->` can be used to exchange dimensional predicate *OldInsInRuralDistrict* for *InsInRuralDistrict* in the slice condition of dimension schema *Insurant*: `trg.broadenSliceCond->(Insurant, OldInsInRuralDistrict, InsInRuralDistrict)`. As dimensional predicate *OldInsInRuralDistrict* is subsumed by dimensional predicate *InsInRuralDistrict*, this navigation step satisfies the operator's precondition. In the case that dimensional predicate *OldInsInRuralDistrict* has to be exchanged by a dimensional predicate, for example, by dimensional predicate *InsInUrbanDistrict*, where both are not related in the subsumption hierarchy, navigation operator `refocusSliceCond->` can be used: `trg.refocusSliceCond->(Insurant, OldInsInRuralDistrict, InsInUrbanDistrict)`.

4.2.4 Operators Changing Base Measure Conditions

Facts of a cube of a non-comparative analysis situation can be restricted by base measure conditions. As for slice conditions, base measure conditions can be narrowed, broadened, or arbitrarily refocused. Narrowing and broadening are done accordingly to the hierarchy of base measure predicates defined in an eDFM. In Table 4.7, six operators used for changing the set of base measure conditions are listed: `narrowBMsrCond+`, `narrowBMsrCond->`, `broadenBMsrCond-`, `broadenBMsrCond->`, `refocusBMsrCond`, and `refocusBMsrCond->`. For navigation operators `narrowBMsrCond+` and `narrowBMsrCond->`, the overall base measure condition of the target analysis situation implies the overall base measure condition of the source, whereas, if operator `broadenBMsrCond-` or `broadenBMsrCond->` is invoked, the overall base measure condition of the source implies the one of the target. Operators `refocusBMsrCond` and `refocusBMsrCond->` do not require any constraints with respect to such implications. The required implications of base measure predicates used in navigation operators `narrowBMsrCond->` and `broadenBMsrCond->` are taken from the hierarchy of base measure predicates defined in the underlying eDFM.

All navigation operators presented in this section involve base measure predicates as parameters. Operator `narrowBMsrCond+` receives a set of base

Table 4.7: Operators narrowBMsCond+, narrowBMsCond->, broadenBMsCond-, broadenBMsCond->, refocusBMsCond, and refocusBMsCond->

Operator Definition	Symbol
<p><u>narrowBMsCond+({B₁, ..., B_b})</u> Precondition: $B_1, \dots, B_b \in \text{BMsPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$, and $\{B_1, \dots, B_b\} \cap \text{BMsConds}_{\text{src}} = \emptyset$. Postcondition: $\text{BMsConds}_{\text{trg}} =$ $\text{BMsConds}_{\text{src}} \cup \{B_1, \dots, B_b\}$</p>	<p>narrowBMsCond+</p> 
<p><u>narrowBMsCond->(B_{old}, B_{new})</u> Precondition: $B_{\text{old}}, B_{\text{new}} \in \text{BMsPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$, $B_{\text{old}} \in \text{BMsConds}_{\text{src}}$, $B_{\text{new}} \notin \text{BMsConds}_{\text{src}}$, and $B_{\text{new}} \Rightarrow B_{\text{old}}$. Postcondition: $\text{BMsConds}_{\text{trg}} =$ $\text{BMsConds}_{\text{src}} \setminus \{B_{\text{old}}\} \cup \{B_{\text{new}}\}$</p>	<p>narrowBMsCond-></p> 
<p><u>broadenBMsCond-({B₁, ..., B_b})</u> Precondition: $\{B_1, \dots, B_b\} \subseteq \text{BMsConds}_{\text{src}}$ Postcondition: $\text{BMsConds}_{\text{trg}} =$ $\text{BMsConds}_{\text{src}} \setminus \{B_1, \dots, B_b\}$</p>	<p>broadenBMsCond-</p> 
<p><u>broadenBMsCond->(B_{old}, B_{new})</u> Precondition: $B_{\text{old}}, B_{\text{new}} \in \text{BMsPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$, $B_{\text{old}} \in \text{BMsConds}_{\text{src}}$, $B_{\text{new}} \notin \text{BMsConds}_{\text{src}}$, and $B_{\text{old}} \Rightarrow B_{\text{new}}$. Postcondition: $\text{BMsConds}_{\text{trg}} =$ $\text{BMsConds}_{\text{src}} \setminus \{B_{\text{old}}\} \cup \{B_{\text{new}}\}$</p>	<p>broadenBMsCond-></p> 
<p><u>refocusBMsCond({B₁, ..., B_b})</u> Precondition: $B_1, \dots, B_b \in \text{BMsPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$ and $\text{BMsConds}_{\text{src}} \neq \{B_1, \dots, B_b\}$. Postcondition: $\text{BMsConds}_{\text{trg}} = \{B_1, \dots, B_b\}$</p>	<p>refocusBMsCond</p> 
<p><u>refocusBMsCond->(B_{old}, B_{new})</u> Precondition: $B_{\text{new}}, B_{\text{old}} \in \text{BMsPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$, $B_{\text{old}} \in \text{BMsConds}_{\text{src}}$, and $B_{\text{new}} \notin \text{BMsConds}_{\text{src}}$. Postcondition: $\text{BMsConds}_{\text{trg}} =$ $\text{BMsConds}_{\text{src}} \setminus \{B_{\text{old}}\} \cup \{B_{\text{new}}\}$</p>	<p>refocusBMsCond-></p> 

measure predicates B_1, \dots, B_b which are added to the set of base measure conditions of the source analysis situation. As a result, the overall base measure condition of the target analysis situation implies the overall base measure condition of the source analysis situation or, in other words, the overall base measure condition of the source analysis situation subsumes the overall base measure condition of the target analysis situation. The application of navigation operator **narrowBMSrCond->** (the second version of narrowing) effects that only one base measure predicate B_{old} of the source analysis situation is exchanged by another base measure predicate B_{new} . As the precondition of this operator requires that B_{new} implies B_{old} , the overall base measure condition of the source analysis situation subsumes the overall base measure condition of the target analysis situation.

Analogously to narrowing, one can broaden the overall base measure condition of a source analysis situation. Navigation operator **broadenBMSrCond-** receives a set of base measure predicates B_1, \dots, B_b as a parameter which represents a subset of base measure conditions of the source analysis situation. This subset of base measure predicates is removed from the set of base measure conditions of the source analysis situation. As a consequence, the overall base measure condition of the target analysis situation subsumes the overall base measure condition of the source. Navigation operator **broadenBMSrCond->** exchanges a base measure predicate B_{old} of the source analysis situation by a base measure predicate B_{new} that is implied by B_{old} .

The last two navigation operators for changing base measure conditions of a source analysis situation allow to make arbitrary changes regardless of implications between the base measure conditions of the source and target analysis situation. Operator **refocusBMSrCond** removes all base measure conditions of the source analysis situation and adds new base measure predicates B_1, \dots, B_b given as a set valued parameter. By navigation operator **refocusBMSrCond->**, a single slice condition B_{old} in the set of base measure conditions of the source analysis situation is exchanged by another base measure predicate B_{new} regardless of possible implications.

In the example depicted in Figure 4.1, operator **narrowBMSrCond+** is applied to source analysis situation **as1** with a set as parameter containing

single base measure predicate *HighCostsPerUnit*. This operator application leads to target analysis situation **as8** that comprises base measure condition *HighCostsPerUnit*, i.e., the target analysis situation is restricted to drug prescriptions having high costs per unit. The new set of base measure conditions of the target implies the old set of the source. Similar to the examples of navigation operations changing slice conditions as presented in Section 4.2.3, one also can find further examples for navigation operators **narrowBMsCond->**, **broadenBMsCond+**, **broadenBMsCond->**, **refocusBMsCond**, and **refocusBMsCond->**.

4.2.5 Operators Changing Aggregate Measures

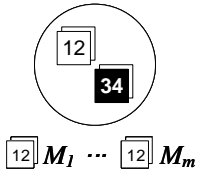
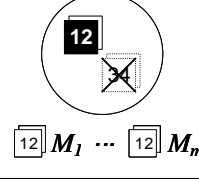
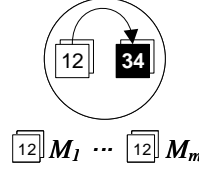
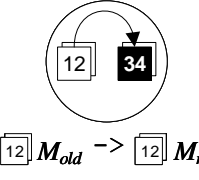
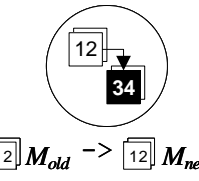
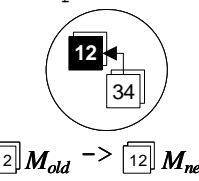
There are five navigation operators changing the set of aggregate measures of a non-comparative analysis situation (Table 4.8): **addMeasure**, **dropMeasure**, **refocusMeasure**, **refocusMeasure->**, **moveDownToMeasure**, and **moveUpToMeasure**.

Operator **addMeasure** receives a set of aggregate measures as a parameter that is added to the set of aggregate measures of a non-comparative analysis situation. Conversely, the set-valued parameter of navigation operator **dropMeasure** represents a set of aggregate measures which are removed from the aggregate measure set of the source analysis situation.

Navigation operator **refocusMeasure** replaces the whole set of aggregate measures of the source analysis situation by the set of aggregate measure given as parameter. There is a second version of refocusing the aggregate measure set of a non-comparative analysis situation (operator **refocusMeasure->**). This operator takes two single aggregate measures as parameters: M_{old} and M_{new} . In this case, aggregate measure M_{old} has to be an element of the aggregate measure set of the source analysis situation which is exchanged by aggregate measure M_{new} .

There are additional navigation operators that also exchange an aggregate measure by another one: **moveDownToMeasure** and **moveUpToMeasure**. In contrast to operator **refocusMeasure->**, both operators use the hierarchy of aggregate measures defined by the sub-aggregate-measure relation. Navi-

Table 4.8: Operators `addMeasure`, `dropMeasure`, `refocusMeasure`, `refocusMeasure->`, `moveDownToMeasure`, and `moveUpToMeasure`

Operator Definition	Symbol
<u>$\text{addMeasure}(\{M_1, \dots, M_m\})$</u> Precondition: $M_1, \dots, M_m \in \text{AMsrs}_C$ with $C = \text{CubeSchema}_{\text{src}}$ and $\{M_1, \dots, M_m\} \cap \text{AMsrs}_{\text{src}} = \emptyset$. Postcondition: $\text{AMsrs}_{\text{trg}} = \text{AMsrs}_{\text{src}} \cup \{M_1, \dots, M_m\}$	<p style="text-align: center;"><code>addMeasure</code></p> 
<u>$\text{dropMeasure}(\{M_1, \dots, M_m\})$</u> Precondition: $\{M_1, \dots, M_m\} \subseteq \text{AMsrs}_{\text{src}}$ Postcondition: $\text{AMsrs}_{\text{trg}} = \text{AMsrs}_{\text{src}} \setminus \{M_1, \dots, M_m\}$	<p style="text-align: center;"><code>dropMeasure</code></p> 
<u>$\text{refocusMeasure}(\{M_1, \dots, M_m\})$</u> Precondition: $M_1, \dots, M_m \in \text{AMsrs}_C$ with $C = \text{CubeSchema}_{\text{src}}$ and $\{M_1, \dots, M_m\} \neq \text{AMsrs}_{\text{src}}$. Postcondition: $\text{AMsrs}_{\text{trg}} = \{M_1, \dots, M_m\}$	<p style="text-align: center;"><code>refocusMeasure</code></p> 
<u>$\text{refocusMeasure->}(M_{\text{old}}, M_{\text{new}})$</u> Precondition: $M_{\text{old}}, M_{\text{new}} \in \text{AMsrs}_C$ with $C = \text{CubeSchema}_{\text{src}}$, $M_{\text{old}} \in \text{AMsrs}_{\text{src}}$, and $M_{\text{new}} \notin \text{AMsrs}_{\text{src}}$. Postcondition: $\text{AMsrs}_{\text{trg}} =$ $\text{AMsrs}_{\text{src}} \setminus \{M_{\text{old}}\} \cup \{M_{\text{new}}\}$	<p style="text-align: center;"><code>refocusMeasure-></code></p> 
<u>$\text{moveDownToMeasure}(M_{\text{old}}, M_{\text{new}})$</u> Precondition: $M_{\text{old}}, M_{\text{new}} \in \text{AMsrs}_C$ with $C = \text{CubeSchema}_{\text{src}}$, $M_{\text{old}} \in \text{AMsrs}_{\text{src}}$, $M_{\text{new}} \notin \text{AMsrs}_{\text{src}}$, and $M_{\text{new}} \twoheadrightarrow M_{\text{old}}$. Postcondition: $\text{AMsrs}_{\text{trg}} = \text{AMsrs}_{\text{src}} \setminus \{M_{\text{old}}\} \cup \{M_{\text{new}}\}$	<p style="text-align: center;"><code>moveDownToMeasure</code></p> 
<u>$\text{moveUpToMeasure}(M_{\text{old}}, M_{\text{new}})$</u> Precondition: $M_{\text{old}}, M_{\text{new}} \in \text{AMsrs}_C$ with $C = \text{CubeSchema}_{\text{src}}$, $M_{\text{old}} \in \text{AMsrs}_{\text{src}}$, $M_{\text{new}} \notin \text{AMsrs}_{\text{src}}$, and $M_{\text{old}} \twoheadrightarrow M_{\text{new}}$. Postcondition: $\text{AMsrs}_{\text{trg}} = \text{AMsrs}_{\text{src}} \setminus \{M_{\text{old}}\} \cup \{M_{\text{new}}\}$	<p style="text-align: center;"><code>moveUpToMeasure</code></p> 

gation operators `moveDownToMeasure` and `moveUpToMeasure` receive two parameters M_{old} and M_{new} such that M_{old} is an element of the aggregate measure set of the source analysis situation whereas aggregate measure M_{new} is not an element of this set. In the case of operator `moveDownToMeasure`, parameter M_{new} represents a sub-aggregate-measure of M_{old} . Conversely, in the case of navigation operator `moveUpToMeasure`, aggregate measure M_{old} represents a sub-aggregate-measure of M_{new} . As for operator `refocusMeasure->`, both navigation operators `moveDownToMeasure` and `moveUpToMeasure` exchanges aggregate measure M_{old} by aggregate measure M_{new} . In contrast to operator `refocusMeasure->`, navigation steps defined by operator `moveDownToMeasure` or `moveUpToMeasure` express additional semantics (moving down or up with respect to the definition of aggregate measures).

Figure 4.1 comprises a navigation step from analysis situation `as1` to `as5` that uses navigation operator `moveUpToMeasure`. In this navigation step, aggregate measure *SumOfCosts* of analysis situation `as1` is replaced by aggregate measure *AvgCostsPerInsurant* yielding analysis situation `as5`, formally written as `as5 = as1.moveUpToMeasure(SumOfCosts, AvgCostsPerInsurant)`. The underlying eDFM of our example (see Figure 2.3) specifies that aggregate measure *SumOfCosts* is used for calculation of aggregate measure *AvgCostsPerInsurant* meaning that *SumOfCosts* is a sub-aggregate-measure of *AvgCostsPerInsurant* (also written as $SumOfCosts \rightarrow AvgCostsPerInsurant$). Thus, in this case, the application of operator `moveUpToMeasure` yields a valid invocation that satisfies the operator's precondition.

By the inverse operator `moveDownToMeasure` (with respect to operator `moveUpToMeasure`), one can navigate from analysis situation `as5` to `as1`: `as1 = as5.moveDownToMeasure(AvgCostsPerInsurant, SumOfCosts)`. If one wants to exchange aggregate measure *SumOfCosts* in analysis situation `as1` by aggregate measure *SumOfQuantity*, navigation operator `refocusMeasure->` can be used: `as1.refocusMeasure->(SumOfCosts, SumOfQuantity)`. Note, in this case, both measures are not related via the sub-aggregate-measure relation. Thus, neither operator `moveDownToMeasure` nor operator `moveUpToMeasure` can be applied.

To exchange the whole aggregate measure set of an analysis situation, op-

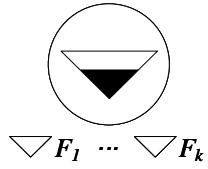
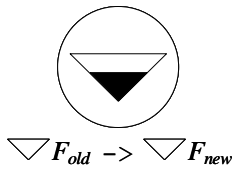
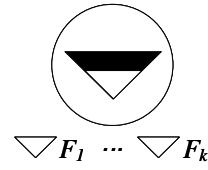
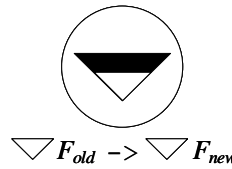
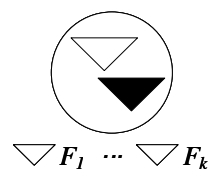
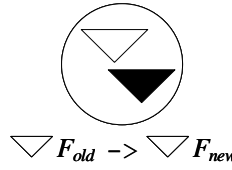
erator `refocusMeasure` can be used. For example, the set of aggregate measures of analysis situation `as1` that only contains aggregate measure `SumOfCosts` can be replaced by a set containing aggregate measures `AvgCostsPerUnit` and `AvgCostsPerInsurant`: `as1.refocusMeasure({AvgCostsPerUnit, AvgCostsPerInsurant})`. The following two concatenated navigation steps yield the same result as in the previous example: `as1.addMeasure({AvgCostsPerUnit, AvgCostsPerInsurant}).dropMeasure({SumOfCosts})`. In this case, aggregate measures `AvgCostsPerUnit` and `AvgCostsPerInsurant` are added by operator `addMeasure`, and, afterwards, aggregate measure `SumOfCosts` is removed by navigation operator `dropMeasure`.

4.2.6 Operators Changing Filter Conditions

The query result of a non-comparative analysis situation can be additionally restricted by filter conditions. Navigation operators changing the set of filter conditions of a non-comparative analysis situation are specified in Table 4.9. They are similar to those changing the set of slice conditions or to those changing the set of base measure conditions. The set of filter conditions can be narrowed, broadened, or arbitrarily refocused. Similar to the set of base measure conditions, narrowing and broadening of filter conditions can be done accordingly to the hierarchy of aggregate measure predicates defined in the underlying eDFM. The application of operators `narrowFilter+` and `narrowFilter->` require that the overall filter condition of the target analysis situation implies the overall filter condition of the source. In contrast, if operators `broadenFilter-` and `broadenFilter->` are invoked, the overall filter condition of the source implies the overall filter condition of the target. Navigation operators `refocusFilter` and `refocusFilter->` do not require any constraints with respect to such implications.

Navigation operators changing filter conditions obtain aggregate measure predicates as parameters. Operator `narrowFilter+` receives a set of aggregate measure predicates F_1, \dots, F_k which are added to the set of filter conditions of the source analysis situation. Thus, the overall filter condition of the source analysis situation subsumes the overall filter condition of the

Table 4.9: Operator $\text{narrowFilter}+$, $\text{narrowFilter}\rightarrow$, $\text{broadenFilter}-$, $\text{broadenFilter}\rightarrow$, refocusFilter , and $\text{refocusFilter}\rightarrow$

Operator Definition	Symbol
$\text{narrowFilter}+(\{F_1, \dots, F_k\})$ Precondition: $F_1, \dots, F_k \in \text{AMsrPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$ and $\{F_1, \dots, F_k\} \cap \text{FilterConds}_{\text{src}} = \emptyset$. Postcondition: $\text{FilterConds}_{\text{trg}} =$ $\text{FilterConds}_{\text{src}} \cup \{F_1, \dots, F_k\}$	$\text{narrowFilter}+$ 
$\text{narrowFilter}\rightarrow(F_{\text{old}}, F_{\text{new}})$ Precondition: $F_{\text{old}}, F_{\text{new}} \in \text{AMsrPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$, $F_{\text{old}} \in \text{FilterConds}_{\text{src}}$, $F_{\text{new}} \notin \text{FilterConds}_{\text{src}}$, and $F_{\text{new}} \Rightarrow F_{\text{old}}$. Postcondition: $\text{FilterConds}_{\text{trg}} =$ $\text{FilterConds}_{\text{src}} \setminus \{F_{\text{old}}\} \cup \{F_{\text{new}}\}$	$\text{narrowFilter}\rightarrow$ 
$\text{broadenFilter}-(\{F_1, \dots, F_k\})$ Precondition: $\{F_1, \dots, F_k\} \subseteq \text{FilterConds}_{\text{src}}$ Postcondition: $\text{FilterConds}_{\text{trg}} =$ $\text{FilterConds}_{\text{src}} \setminus \{F_1, \dots, F_k\}$	$\text{broadenFilter}-$ 
$\text{broadenFilter}\rightarrow(F_{\text{old}}, F_{\text{new}})$ Precondition: $F_{\text{old}}, F_{\text{new}} \in \text{AMsrPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$, $F_{\text{old}} \in \text{FilterConds}_{\text{src}}$, $F_{\text{new}} \notin \text{FilterConds}_{\text{src}}$, and $F_{\text{old}} \Rightarrow F_{\text{new}}$. Postcondition: $\text{FilterConds}_{\text{trg}} =$ $\text{FilterConds}_{\text{src}} \setminus \{F_{\text{old}}\} \cup \{F_{\text{new}}\}$	$\text{broadenFilter}\rightarrow$ 
$\text{refocusFilter}(\{F_1, \dots, F_k\})$ Precondition: $F_1, \dots, F_k \in \text{AMsrPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$ and $\text{FilterConds}_{\text{src}} \cap \{F_1, \dots, F_k\} = \emptyset$. Postcondition: $\text{FilterConds}_{\text{trg}} = \{F_1, \dots, F_k\}$	refocusFilter 
$\text{refocusFilter}\rightarrow(F_{\text{old}}, F_{\text{new}})$ Precondition: $F_{\text{old}}, F_{\text{new}} \in \text{MsrPredicates}_C$ with $C = \text{CubeSchema}_{\text{src}}$, $F_{\text{old}} \in \text{FilterConds}_{\text{src}}$, and $F_{\text{new}} \notin \text{FilterConds}_{\text{src}}$. Postcondition: $\text{FilterConds}_{\text{trg}} =$ $\text{FilterConds}_{\text{src}} \setminus \{F_{\text{old}}\} \cup \{F_{\text{new}}\}$	$\text{refocusFilter}\rightarrow$ 

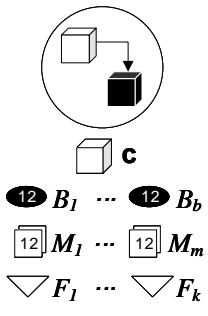
target analysis situation. Navigation operator `narrowFilter->` can be used, if only one aggregate measure predicate F_{old} of the source analysis situation has to be exchanged by another aggregate measure predicate F_{new} . The precondition of this operator requires that F_{new} implies F_{old} .

Similar to operators changing base measure conditions, navigation operator `broadenFilter-` removes aggregate measure predicates F_1, \dots, F_k (given as a set parameter) from the set of filter conditions and operator `broadenFilter->` exchanges aggregate measure predicate F_{old} by F_{new} (required that F_{new} is implied by F_{old}). In both cases, the overall filter condition of the target analysis situation subsumes the overall filter condition of the source.

If there are no implications between the overall filter conditions of the source and the target, one can use navigation operators `refocusFilter` and `refocusFilter->`. In the first case, the whole set of filter conditions of the source is replaced by the set of aggregate measure predicates F_1, \dots, F_k given as a set-valued parameter. In contrast, operator `refocusFilter->` exchanges aggregate measure predicate F_{old} by F_{new} . Note that in this case, no implication between F_{old} and F_{new} is required.

In our example of Figure 4.1, aggregate measure predicate *HighAvgDrugPrescrCostsPerIns* is removed from the set of filter conditions of analysis situation `as1` by operator `broadenFilter-` yielding target analysis situation `as6` that does not contain filter conditions any more: `as6 = as1.broadenFilter-({HighAvgDrugPrescrCostsPerIns})`. In the query execution of source analysis situation `as1` only records with high average drug prescription costs per insurant are returned, whereas in the target `as6` all records without restrictions are obtained. Further examples using navigation operators that change the set of filter conditions of a non-comparative analysis situation can be constructed similar to those examples presented in the previous sections which demonstrate operators changing slice conditions and base measure conditions.

Table 4.10: Operator `drillAcrossToCube`

Operator Definition	Symbol
<p>$\text{drillAcrossToCube}(\mathbf{c}, \{B_1, \dots, B_b\}, \{M_1, \dots, M_m\}, \{F_1, \dots, F_k\})$</p> <hr/> <p>Precondition: \mathbf{c} is a cube instance with $C = \text{CubeSchema}_{\mathbf{c}}$ and $\mathbf{c} \neq \text{CubeInstance}_{\text{src}}$, $B_1, \dots, B_b \in \text{BMSrPredicates}_C$, $M_1, \dots, M_m \in \text{AMsrPredicates}_C$, $\{M_1, \dots, M_m\} \neq \emptyset$, and $F_1, \dots, F_k \in \text{AMsrPredicates}_C$.</p> <p>Postcondition: $\text{CubeInstance}_{\text{trg}} = \mathbf{c}$, $\text{BMSrConds}_{\text{trg}} = \{B_1, \dots, B_b\}$, and if $D \in \text{DimSchemas}_{\text{src}} \cap \text{DimSchemas}_C$, then $\text{DimQual}_{\text{src}}(D) \in \text{DimQuals}_{\text{trg}}$, if $D \in \text{DimSchemas}_C \wedge D \notin \text{DimSchemas}_{\text{src}}$, then $\text{DiceLvl}_{\text{trg}}(D) = \text{top}_D$, $\text{DiceNode}_{\text{trg}}(D) = \mathbf{all}_{\mathbf{d}}$ with $\mathbf{d} = \text{DimInstance}_{\text{trg}}(D)$, $\text{SliceConds}_{\text{trg}}(D) = \emptyset$, and $\text{GranLvl}_{\text{trg}}(D) = \text{top}_D$, and, moreover, $\text{AMsr}_{\text{trg}} = \{M_1, \dots, M_m\}$ and $\text{FilterConds}_{\text{trg}} = \{F_1, \dots, F_k\}$.</p>	<p style="text-align: center;"><code>drillAcrossToCube</code></p>  <p style="text-align: center;"> \mathbf{c} $\textcircled{12} B_1 \dots \textcircled{12} B_b$ $\boxed{12} M_1 \dots \boxed{12} M_m$ $\nabla F_1 \dots \nabla F_k$ </p>

4.2.7 Operator Changing Cube Access

Navigation operator `drillAcrossToCube` specified in Table 4.10 changes the cube instance of a non-comparative analysis situation. In most cases, this operator exchanges the cube instance of the source by a cube instance of a different cube schema. This is the most profound change of an analysis situation. For example, one wants to analyze drug prescriptions and in the next situation she or he wants to analyze ambulant treatments or hospitalizations. In this case, other business events (facts) are explored. For instance, using operator `drillAcrossToCube`, one navigates from an analysis situation comprising an instance of cube schema *DrugPrescription* to an analysis situation containing an instance of cube schema *AmbTreatment* or *Hospitalization*.

Operator `drillAcrossToCube` takes four parameters: one parameter that denotes the new cube instance `c` to be queried, a possible empty set of base measure predicates B_1, \dots, B_b that becomes the set of base measure conditions, a non-empty set of aggregate measures M_1, \dots, M_m that becomes the set of aggregate measures of the target analysis situation, and a possible empty set of aggregate measure predicates F_1, \dots, F_k that becomes the set of filter conditions. These parameters specify the constituents (apart from dimension qualifications) of target analysis situation.

Dimension qualifications are transferred from the source to the target analysis situation as far as it is possible with respect to the cube instance of the target. This means that dimension qualifications in the source that do not fit to the schema of the cube instance of the target are omitted. For dimension schemas that are in the schema of the cube instance of the target but not in the schema of the cube instance of the source, a default dimension qualification for the target analysis situation is constructed with dice level *top*, dice node *all*, \emptyset as the set of slice conditions, and granularity level *top*. Finally, dimension qualifications of the source analysis situation that also can be applied to the target are also constituents of the target analysis situation.

In the example of Figure 4.1, analysis situation `as6` is linked to `as7` by navigation operator `drillAcrossToCube`: `as7 = as6.drillAcrossToCube(Hospitalization, \emptyset , {AvgCostsPerInsurant, AvgCostsPerDay}, \emptyset)`. Cube schema

DrugPrescription of source *as6* is replaced by cube schema *Hospitalization* in the target analysis situation *as7*. Two measures *AvgCostsPerInsurant* and *AvgCostsPerDay* (both defined on cube schema *Hospitalization*) are assigned to analysis situation *as7*. There are no base measure conditions and filter conditions for analysis situation *as7*. Thus, empty sets for base measure predicates and aggregate measure predicates are given as parameters. Because both dimension qualifications of the source with respect to dimension schemas *Time* and *Insurant* are also applicable on cube schema *Hospitalization*, these dimension qualifications are also present in analysis situation *as7*. Dimension qualifications with respect to dimension schemas *Drug* and *Doctor* (which are applicable on source analysis situation *as6*) must be omitted because this dimension schemas are not applicable on analysis situation *as7*. For dimension schema *Hospital* which is a part of cube schema *Hospitalization*, a default dimension qualification (with dice level *top*, dice node *all*, \emptyset as the set of slice conditions, and granularity level *top*) is assigned to target analysis situation *as7*.

Navigation operator *drillAcrossToCube* was the last one we presented in the group of operators that only involve non-comparative analysis situations (for both source analysis situation and target analysis situation). In the next section, we present navigation operators involving comparative analysis situations.

4.3 Operators Involving Comparison

This section introduces navigation operators that involve comparative analysis situations (comparative navigation operators). We distinguish three general groups of such navigation operators: (1) navigation operators introducing comparison (Tables 4.11 and 4.12), (2) navigation operators changing comparison (Tables 4.13–4.19), and (3) navigation operators dropping comparisons (Table 4.20). The first operator group takes a non-comparative analysis situation and generates a comparative one, i.e., these operators introduce comparisons. The second group has a comparative analysis situation for both the source analysis situation and the target analysis situation, i.e.,

a comparative analysis situation is modified meaning comparison is changed. Finally, a user can take a comparative analysis situation, drop comparison, and analyze either the context of interest or the context of comparison, i.e., she or he navigates to a non-comparative analysis situation. Figures 4.2–4.7 demonstrate examples of navigation steps with operators involving comparative analysis situations.

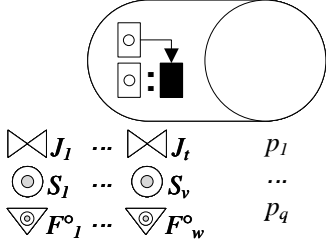
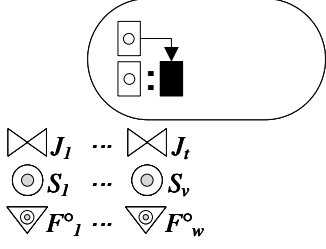
4.3.1 Operators Introducing Comparison

In this subsection, navigation operators that have a non-comparative analysis situation as source and a comparative one as target are presented. Both the context of interest and the context of comparison are derived from the non-comparative analysis situation. As a precondition, we suppose that **src** is the non-comparative source analysis situation and **trg** is the comparative target analysis situation. Furthermore, we assume that *OP* is a navigation operator only involving non-comparative analysis situations. Operator *OP* specifies how either the context of comparison or the context of interest is generated from the non-comparative source analysis situation.

First variant of operator **relate** in Table 4.11 introduces a comparison in the way that the context of comparison is derived from the non-comparative source analysis situation **src** by applying operator *OP* that generates target $CoC_{\mathbf{trg}}$, i.e., $CoC_{\mathbf{trg}} = \mathbf{src}.OP(p_1, \dots, p_q)$. Parameters p_1, \dots, p_q represent the actual parameters of operator *OP*. The constituents of the original non-comparative analysis situation schema become the context of interest, i.e., $CoI_{\mathbf{trg}} = \mathbf{src}$. The set of scores, the set of join conditions, and the set of score filter conditions of the comparative target analysis situation **trg** are specified as additional parameters of operator **relate**. The empty circle of the pictogram of **relate** has to be filled with the pictogram of the operator *OP*.

Figure 4.2 shows an example of a navigation step containing operator **relate**. In the source analysis situation **as1**, drug prescription costs of year 2016 are shown per insurants' province. If a user wants to compare them with the previous year 2015 by calculating the ratio of costs, one trans-

Table 4.11: Operator relate

Operator Definition	Symbol
<p> $\text{relate}(OP(p_1, \dots, p_q), \{J_1, \dots, J_t\}, \{S_1, \dots, S_v\}, \{F_1^\circ, \dots, F_w^\circ\})$ </p> <p> Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situation src; J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $DimSchemas_{\text{src}.OP(p_1, \dots, p_q)}$; S_1, \dots, S_v are scores defined over aggregate measures in $AMsr_{\text{src}.OP(p_1, \dots, p_q)}$; $F_1^\circ, \dots, F_w^\circ$ are score predicates defined over $\{S_1, \dots, S_v\}$ and $AMsr_{\text{src}.OP(p_1, \dots, p_q)}$. </p> <p> Postcondition: $CoI_{\text{trg}} = \text{src}$, $CoC_{\text{trg}} = \text{src}.OP(p_1, \dots, p_q)$, $JoinConds_{\text{trg}} = \{J_1, \dots, J_t\}$, $Scores_{\text{trg}} = \{S_1, \dots, S_v\}$, and $ScoreFilters_{\text{trg}} = \{F_1^\circ, \dots, F_w^\circ\}$. </p>	<p style="text-align: center;">relate OP</p>  <p style="text-align: center;"> $\boxtimes J_1 \dots \boxtimes J_t \quad p_1$ $\odot S_1 \dots \odot S_v \quad \dots$ $\nabla F_1^\circ \dots \nabla F_w^\circ \quad p_1$ </p>
<p> $\text{relate}(\{J_1, \dots, J_t\}, \{S_1, \dots, S_v\}, \{F_1^\circ, \dots, F_w^\circ\})$ </p> <p> Precondition: J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $DimSchemas_{\text{src}}$; S_1, \dots, S_v are scores defined over aggregate measures in $AMsr_{\text{src}}$; and $F_1^\circ, \dots, F_w^\circ$ are score predicates defined over $\{S_1, \dots, S_v\}$ and $AMsr_{\text{src}}$. </p> <p> Postcondition: $CoI_{\text{trg}} = \text{src}$, $CoC_{\text{trg}} = \text{src}$, $JoinConds_{\text{trg}} = \{J_1, \dots, J_t\}$, $Scores_{\text{trg}} = \{S_1, \dots, S_v\}$, and $ScoreFilters_{\text{trg}} = \{F_1^\circ, \dots, F_w^\circ\}$. </p>	<p style="text-align: center;">relate</p>  <p style="text-align: center;"> $\boxtimes J_1 \dots \boxtimes J_t$ $\odot S_1 \dots \odot S_v$ $\nabla F_1^\circ \dots \nabla F_w^\circ$ </p>

fers *as1* as context of interest, derives a context of comparison by applying operator *moveToPrevNode* in dimension *Time*, and joins both by condition *SameInsProvince*. Score *RatioOfSumOfCosts* is used as a measure of comparison. Both join condition *SameInsProvince* and score predicate *RatioOfSumOfCosts* are defined in the eDFM of Figure 2.3. On the left side of the navigation operator symbol, one can recognize the name of operator *relate*, the symbolization of operator *relate*, the set of join conditions containing the single element *SameInsProvince*, and the set of scores with single element *RatioOfSumOfCosts*. Also the set of score predicates would be positioned on the left side but in our example we have an empty set. Operator *moveToPrevNode* is invoked to generate the context of comparison. The name, the symbol, and the parameters (in this case dimension schema *Time*) of operator *moveToPrevNode* are depicted on the right side. In a formal notation we can write this navigation step in the following way: $as2 = as1.relate(moveToPrevNode(Time), \{SameInsProvince\}, \{RatioOfSumOfCosts\}, \{\})$.

The second variant of operator *relate* has no non-comparative operator *OP*. It can be used to join a non-comparative source analysis situation with itself, i.e., it can be used, if one intends to introduce comparison at granularity level. Figure 4.3 presents an example comprising a non-comparative source analysis situation *as1* with granularity level *year* in the dimension qualification of dimension schema *Time* and granularity level *insProvince* with respect to dimension schema *Insurant*. The comparative target analysis situation *as2* of this navigation step comprises source *as1* for both the context of interest and the context of comparison which are joined by join conditions *SameInsProvince* with respect to dimension schema *Insurant* and *PrevYear* with respect to dimension schema *Time*. Both join conditions are defined in the eDFM presented in Figure 2.3. Whereas join condition *SameInsProvince* links equal insurants' province, join condition *PrevYear* joins a year with its previous year. The translation into SQL specifies inner joins for join conditions. Thus, a year will be only listed, if there also exists a previous year in the corresponding time dimension. In the target analysis situation of this example, drug prescription costs of a province are listed per

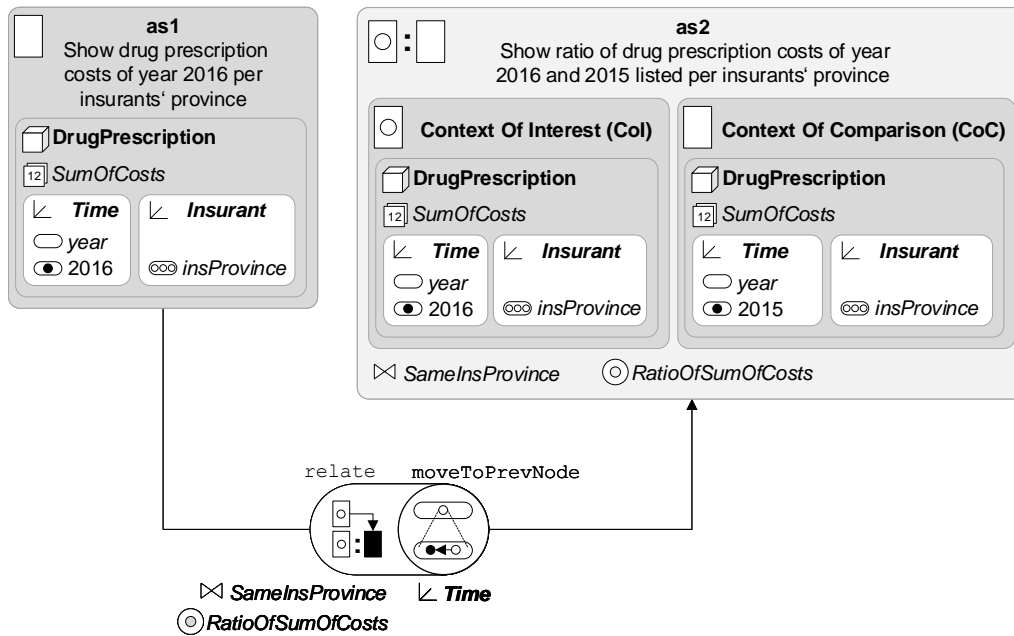


Figure 4.2: Example of a comparative navigation step containing first variant of operator *relate*

year and in each result row costs are compared with the previous year. As a measure of comparison, the ratio of drug prescription costs specified by score *RatioOfSumOfCosts* is calculated. In this example, navigation operator *relate* comprises join conditions *SameInsProvince* and *PrevYear*, and score *RatioOfSumOfCosts* as actual parameters. Note that there is no non-comparative navigation operator. The navigation step of Figure 4.3 can be written formally in the following way: $as2 = as1.\text{relate}(\{SameInsProvince, PrevYear\}, \{RatioOfSumOfCosts\}, \{\})$. In this example, again there is no score predicate. Thus, the third actual parameter is an empty set.

Navigation operator *target* specified in Table 4.12 is another example that comprises a non-comparative analysis situation as source and a comparative one as target. Whereas operator *relate* derives the context of comparison from the non-comparative source analysis situation, navigation operator *target* derives the context of interest. The constituents of the source analysis situation are transferred unmodified to the context of comparison. For

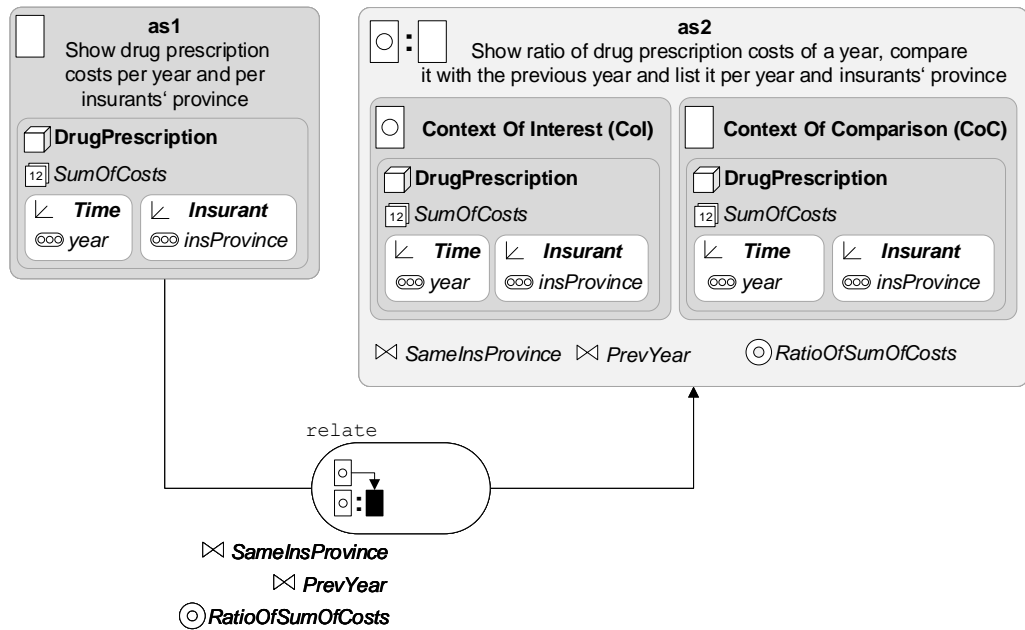
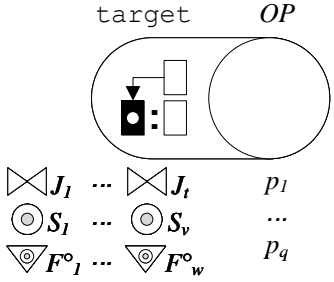


Figure 4.3: Example of a comparative navigation step containing second variant of operator **relate**

the context of interest, non-comparative operator OP is applied to the non-comparative source analysis situation. The join of a non-comparative source analysis situation with itself can be accomplished by the second variant of operator **relate**. Thus, there is no need to define a second variant for navigation operator **target**.

In Figure 4.4, a navigation step (using operator **target**) is shown that yields the same comparative target analysis situation as attained in the example depicted in Figure 4.2. As a difference, in the source analysis situation of Figure 4.4, the user analyzes drug prescription costs of year 2015 and navigates to the target analysis situation to compare with year 2016. In this case, the context of interest is changed (with respect to the non-comparative source analysis situation) to dice node 2016 by non-comparative navigation operator **moveToNextNode**. Formally, this navigation step can be written as $as2 = as1.target(moveToNextNode(Time), \{SameInsProvince\}, \{RatioOfSumOfCosts\}, \{\})$.

Table 4.12: Operator target

Operator Definition	Symbol
<p> $\text{target}(OP(p_1, \dots, p_q), \{J_1, \dots, J_t\}, \{S_1, \dots, S_v\}, \{F_1^\circ, \dots, F_w^\circ\})$ </p> <hr/> <p> Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situation \mathbf{src}; J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $DimSchemas_{\mathbf{src}.OP(p_1, \dots, p_q)}$; S_1, \dots, S_v are scores defined over aggregate measures in $AMsrs_{\mathbf{src}.OP(p_1, \dots, p_q)}$; $F_1^\circ, \dots, F_w^\circ$ are score predicates defined over $\{S_1, \dots, S_v\}$ and $AMsrs_{\mathbf{src}.OP(p_1, \dots, p_q)}$. </p> <p> Postcondition: $CoI_{\mathbf{trg}} = \mathbf{src}.OP(p_1, \dots, p_q)$, $CoC_{\mathbf{trg}} = \mathbf{src}$, $JoinConds_{\mathbf{trg}} = \{J_1, \dots, J_t\}$, $Scores_{\mathbf{trg}} = \{S_1, \dots, S_v\}$, and $ScoreFilters_{\mathbf{trg}} = \{F_1^\circ, \dots, F_w^\circ\}$. </p>	<div style="text-align: center;"> <p>target OP</p>  </div>

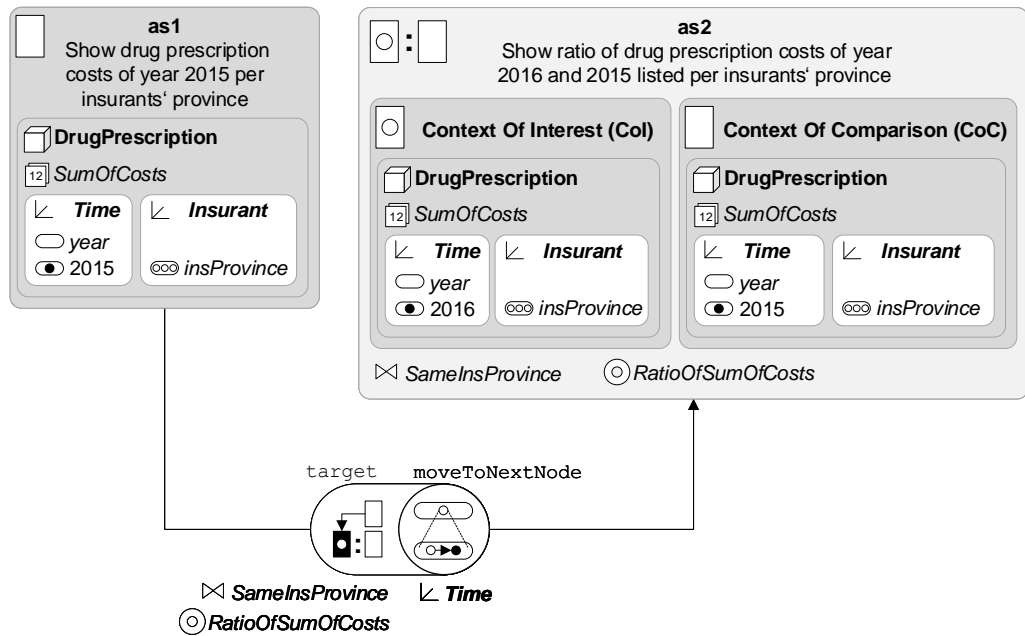


Figure 4.4: Example of a comparative navigation step containing operator target

4.3.2 Operators Changing Comparison

Comparative analysis situations can be modified by navigation operators defined in Tables 4.13–4.19. These operators take a comparative source analysis situation and return a comparative target analysis situation derived from the source.

Navigation operators **rerelate** (see Table 4.13), **retarget** (see Table 4.14), and **correlate** (see Table 4.15 and 4.16) have another navigation operator OP as parameter with a non-comparative analysis situation for both source and target. Operator **rerelate** is applied to CoC_{src} , modifies it, and returns the non-comparative analysis situation for CoC_{trg} . Analogously, operator **retarget** is applied to CoI_{src} , again modifies it, and returns the non-comparative analysis situation for CoI_{trg} . Operator **correlate** modifies both the context of interest and the context of comparison by OP , i.e., $CoI_{trg} = CoI_{src}.OP(p_1, \dots, p_q)$ and $CoC_{trg} = CoC_{src}.OP(p_1, \dots, p_q)$.

Operators **rerelate**, **retarget**, and **correlate** are available in three

variants. In the first variant, additional parameters are defined to receive a new set of join conditions J_1, \dots, J_t , a new set of scores S_1, \dots, S_v , and a new set of score filters $F_1^\circ, \dots, F_w^\circ$ for the target analysis situation. In the second variant, only a new set of join conditions has to be indicated. This variant is useful, if non-comparative navigation operator OP changes the granularity level of the context of interest and/or of the context of comparison—in this case, the join condition has to be redefined. The third variant only has a non-comparative operator OP as a parameter. Join conditions, scores, and score filters remain unchanged.

Figure 4.5 shows an example of a comparative navigation step using operator `rerelate`. The context of comparison is changed with respect to the dice node (concerning dimension schema *Time*) by non-comparative operator `moveToPrevNode`. Whereas in the comparative source analysis situation year 2016 (context of interest) is compared with year 2015 (context of comparison), in the target analysis situation, the dice node of the context of comparison is changed to year 2014 by operator `moveToPrevNode`, i.e., in the target, year 2016 is compared with year 2014. Formally, the whole navigation step can be written as: `as2 = as1.rerelate(moveToPrevNode(Time))`. In this example, the third variant of operator `rerelate` is applied without redefining join conditions, scores, and score filters. Similarly, the context of interest can be changed by navigation operator `retarget`.

In Figure 4.6, three navigation steps are depicted. Two of them include operator `correlate`. Navigation step from comparative source analysis situation `as1` to comparative target analysis situation `as2` changes both context of interest and context of comparison. In this example, non-comparative operator `narrowSliceCond+` is used to narrow slice condition in the target with respect to the dimension qualification of dimension schema *Insurant*. For both CoI_{as2} and CoC_{as2} comparison of insurants' provinces is restricted to rural districts by adding slice condition *InsInRuralDistrict*. The whole comparative navigation step can be written formally as `as2 = as1.correlate(narrowSliceCond+(Insurant, InsInRuralDistrict))`.

In the second navigation step of Figure 4.6 from analysis situation `as2` to analysis situation `as3`, a correlated change in target `as3` is performed by ap-

Table 4.13: Operator `rerelate`

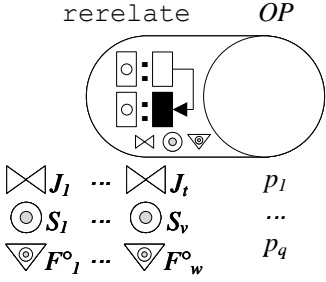
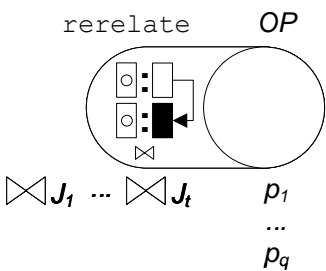
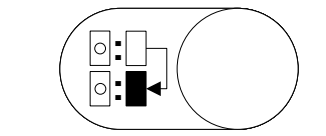
Operator Definition	Symbol
<p>$\text{rerelate}(OP(p_1, \dots, p_q), \{J_1, \dots, J_t\}, \{S_1, \dots, S_v\}, \{F_1^\circ, \dots, F_w^\circ\})$</p> <p>Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situation CoC_{src}; J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $DimSchemas_{CoI_{\text{src}}}$ and $DimSchemas_{CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)}$; S_1, \dots, S_v are scores defined over aggregate measures in $AMsrs_{CoI_{\text{src}}}$ and $AMsrs_{CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)}$; $F_1^\circ, \dots, F_w^\circ$ are score predicates defined over $\{S_1, \dots, S_v\}$, $AMsrs_{CoI_{\text{src}}}$, and $AMsrs_{CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)}$.</p> <p>Postcondition: $CoC_{\text{trg}} = CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)$, $JoinConds_{\text{trg}} = \{J_1, \dots, J_t\}$, $Scores_{\text{trg}} = \{S_1, \dots, S_v\}$, and $ScoreFilters_{\text{trg}} = \{F_1^\circ, \dots, F_w^\circ\}$.</p>	<p style="text-align: center;">rerelate OP</p>  <p style="text-align: right;">p_1 ... p_q</p>
<p>$\text{rerelate}(OP(p_1, \dots, p_q), \{J_1, \dots, J_t\})$</p> <p>Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situation CoC_{src}; J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $DimSchemas_{CoI_{\text{src}}}$ and $DimSchemas_{CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)}$.</p> <p>Postcondition: $CoC_{\text{trg}} = CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)$ and $JoinConds_{\text{trg}} = \{J_1, \dots, J_t\}$.</p>	<p style="text-align: center;">rerelate OP</p>  <p style="text-align: right;">p_1 ... p_q</p>
<p>$\text{rerelate}(OP(p_1, \dots, p_q))$</p> <p>Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situation CoC_{src}.</p> <p>Postcondition: $CoC_{\text{trg}} = CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)$</p>	<p style="text-align: center;">rerelate OP</p>  <p style="text-align: right;">p_1 ... p_q</p>

Table 4.14: Operator retarget

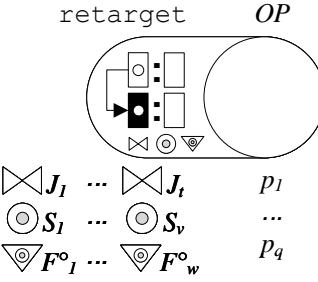
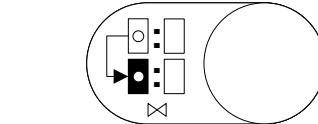
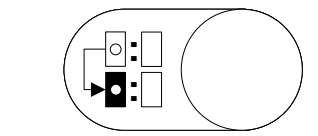
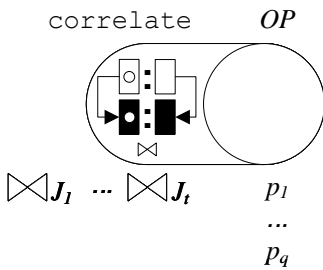
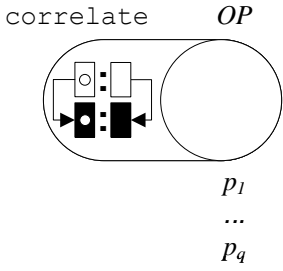
Operator Definition	Symbol
<p>$\text{retarget}(OP(p_1, \dots, p_q), \{J_1, \dots, J_t\}, \{S_1, \dots, S_v\}, \{F_1^\circ, \dots, F_w^\circ\})$</p> <p>Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situation CoI_{src}; J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $DimSchemas_{CoC_{\text{src}}}$ and $DimSchemas_{CoI_{\text{src}}.OP(p_1, \dots, p_q)}$; S_1, \dots, S_v are scores defined over aggregate measures in $AMsrs_{CoC_{\text{src}}}$ and $AMsrs_{CoI_{\text{src}}.OP(p_1, \dots, p_q)}$; $F_1^\circ, \dots, F_w^\circ$ are score predicates defined over $\{S_1, \dots, S_v\}$, $AMsrs_{CoC_{\text{src}}}$, and $AMsrs_{CoI_{\text{src}}.OP(p_1, \dots, p_q)}$.</p> <p>Postcondition: $CoI_{\text{trg}} = CoI_{\text{src}}.OP(p_1, \dots, p_q)$, $JoinConds_{\text{trg}} = \{J_1, \dots, J_t\}$, $Scores_{\text{trg}} = \{S_1, \dots, S_v\}$, and $ScoreFilters_{\text{trg}} = \{F_1^\circ, \dots, F_w^\circ\}$.</p>	<p style="text-align: center;">retarget OP</p>  <p style="text-align: center;"> $\times J_1 \dots \times J_t$ p_1 $\odot S_1 \dots \odot S_v$ \dots $\nabla F_1^\circ \dots \nabla F_w^\circ$ p_q </p>
<p>$\text{retarget}(OP(p_1, \dots, p_q), \{J_1, \dots, J_t\})$</p> <p>Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situation CoI_{src}; J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $DimSchemas_{CoC_{\text{src}}}$ and $DimSchemas_{CoI_{\text{src}}.OP(p_1, \dots, p_q)}$.</p> <p>Postcondition: $CoI_{\text{trg}} = CoI_{\text{src}}.OP(p_1, \dots, p_q)$ and $JoinConds_{\text{trg}} = \{J_1, \dots, J_t\}$.</p>	<p style="text-align: center;">retarget OP</p>  <p style="text-align: center;"> $\times J_1 \dots \times J_t$ p_1 \dots p_q </p>
<p>$\text{retarget}(OP(p_1, \dots, p_q))$</p> <p>Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situation CoI_{src}.</p> <p>Postcondition: $CoI_{\text{trg}} = CoI_{\text{src}}.OP(p_1, \dots, p_q)$</p>	<p style="text-align: center;">retarget OP</p>  <p style="text-align: center;"> p_1 \dots p_q </p>

Table 4.15: Operator correlate (part 1)

Operator Definition	Symbol
<p> $\text{correlate}(OP(p_1, \dots, p_q), \{J_1, \dots, J_t\}, \{S_1, \dots, S_v\}, \{F_1^\circ, \dots, F_w^\circ\})$ </p> <hr/> <p> Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situations CoI_{src} and CoC_{src}; J_1, \dots, J_t are join conditions defined over dimension schemas in $DimSchemas_{CoI_{\text{src}}.OP(p_1, \dots, p_q)}$ and $DimSchemas_{CoC_{\text{src}}.OP(p_1, \dots, p_q)}$; S_1, \dots, S_v are scores defined over aggregate measures in $AMsrs_{CoI_{\text{src}}.OP(p_1, \dots, p_q)}$ and $AMsrs_{CoC_{\text{src}}.OP(p_1, \dots, p_q)}$; $F_1^\circ, \dots, F_w^\circ$ are score predicates defined over $\{S_1, \dots, S_v\}$, $AMsrs_{CoI_{\text{src}}.OP(p_1, \dots, p_q)}$, and $AMsrs_{CoC_{\text{src}}.OP(p_1, \dots, p_q)}$. </p> <p> Postcondition: $CoI_{\text{trg}} = CoI_{\text{src}}.OP(p_1, \dots, p_q)$, $CoC_{\text{trg}} = CoC_{\text{src}}.OP(p_1, \dots, p_q)$, $JoinConds_{\text{trg}} = \{J_1, \dots, J_t\}$, $Scores_{\text{trg}} = \{S_1, \dots, S_v\}$, and $ScoreFilters_{\text{trg}} = \{F_1^\circ, \dots, F_w^\circ\}$. </p>	<p style="text-align: center;">correlate OP</p> <p style="text-align: center;"> $\square J_1 \dots \square J_t \quad p_1$ $\odot S_1 \dots \odot S_v \quad \dots$ $\triangledown F_1^\circ \dots \triangledown F_w^\circ \quad p_q$ </p>

Table 4.16: Operator correlate (part 2)

Operator Definition	Symbol
<p>$\text{correlate}(OP(p_1, \dots, p_q), \{J_1, \dots, J_t\})$</p> <p>Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situations CoI_{src} and CoC_{src}; J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $DimSchemas_{CoI_{\text{src}} \cdot OP(p_1, \dots, p_q)}$ and $DimSchemas_{CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)}$.</p> <p>Postcondition: $CoI_{\text{trg}} = CoI_{\text{src}} \cdot OP(p_1, \dots, p_q)$, $CoC_{\text{trg}} = CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)$, and $JoinConds_{\text{trg}} = \{J_1, \dots, J_t\}$.</p>	 <p>The symbol consists of a large circle labeled "OP" on the right. To its left is a smaller circle labeled "correlate" containing a diagram with two vertical rectangles, a central vertical line with a dot, and arrows. Below this diagram are join symbols $\bowtie J_1 \dots \bowtie J_t$ and parameters $p_1 \dots p_q$.</p>
<p>$\text{correlate}(OP(p_1, \dots, p_q))$</p> <p>Precondition: OP is a non-comparative navigation operator (with p_1, \dots, p_q as actual parameters) that can be applied to non-comparative analysis situations CoI_{src} and CoC_{src}.</p> <p>Postcondition: $CoI_{\text{trg}} = CoI_{\text{src}} \cdot OP(p_1, \dots, p_q)$ and $CoC_{\text{trg}} = CoC_{\text{src}} \cdot OP(p_1, \dots, p_q)$.</p>	 <p>The symbol consists of a large circle labeled "OP" on the right. To its left is a smaller circle labeled "correlate" containing a diagram with two vertical rectangles, a central vertical line with a dot, and arrows. Below this diagram are parameters $p_1 \dots p_q$.</p>

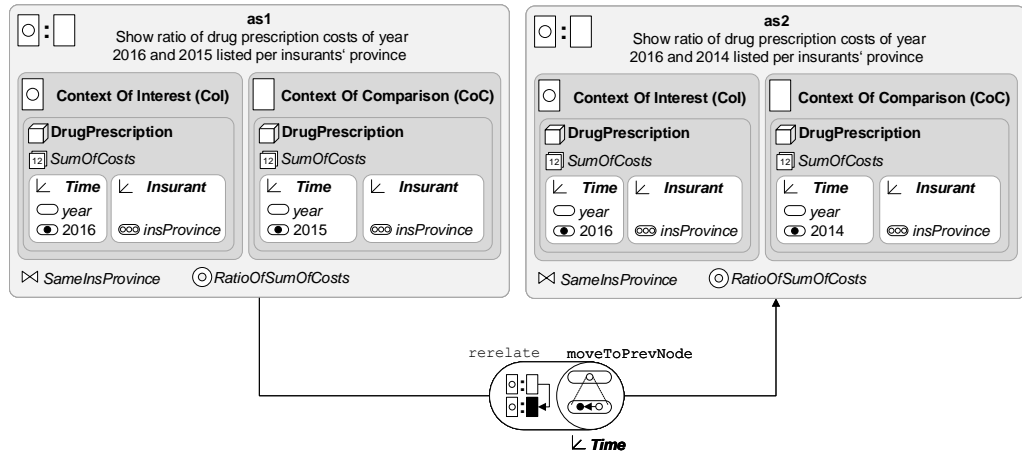


Figure 4.5: Example of a comparative navigation step containing a variant of operator `rerelate`

plying operator `drillDownToLevel` to both CoI_{as2} and CoC_{as2} . As a result, granularity levels of CoI_{as3} and CoC_{as3} are refined respectively to dimension level $insDistrict$. In this case, also the join condition has to be changed from $SameInsProvince$ in source $as2$ to $SameInsDistrict$ in target $as3$. Note, that join condition $SameInsDistrict$ is not depicted in the eDFM of Figure 2.3 due to space restrictions. Analogously to join condition $SameInsProvince$, join condition $SameInsDistrict$ can be defined with respect to dimension level $insDistrict$ by expression $CoI.insDistrict = CoC.insDistrict$. Formally, the whole navigation step can be written in the following way: $as3 = as2.correlate(drillDownToLevel(Insurant, insDistrict), SameInsDistrict)$.

In Tables 4.17 and 4.18, navigation operators are specified that modify score filters of a comparative analysis situation. These operators work similarly to those navigation operators which change filter conditions of a non-comparative analysis situation. Navigation operator `narrowScoreFilter+` adds additional score predicates. As a consequence, the overall score filter of the comparative source analysis situation subsumes the overall score filter of the target. Operator `narrowScoreFilter->` can be used, if a single score predicate of the score filter of the source analysis situation has to be exchanged

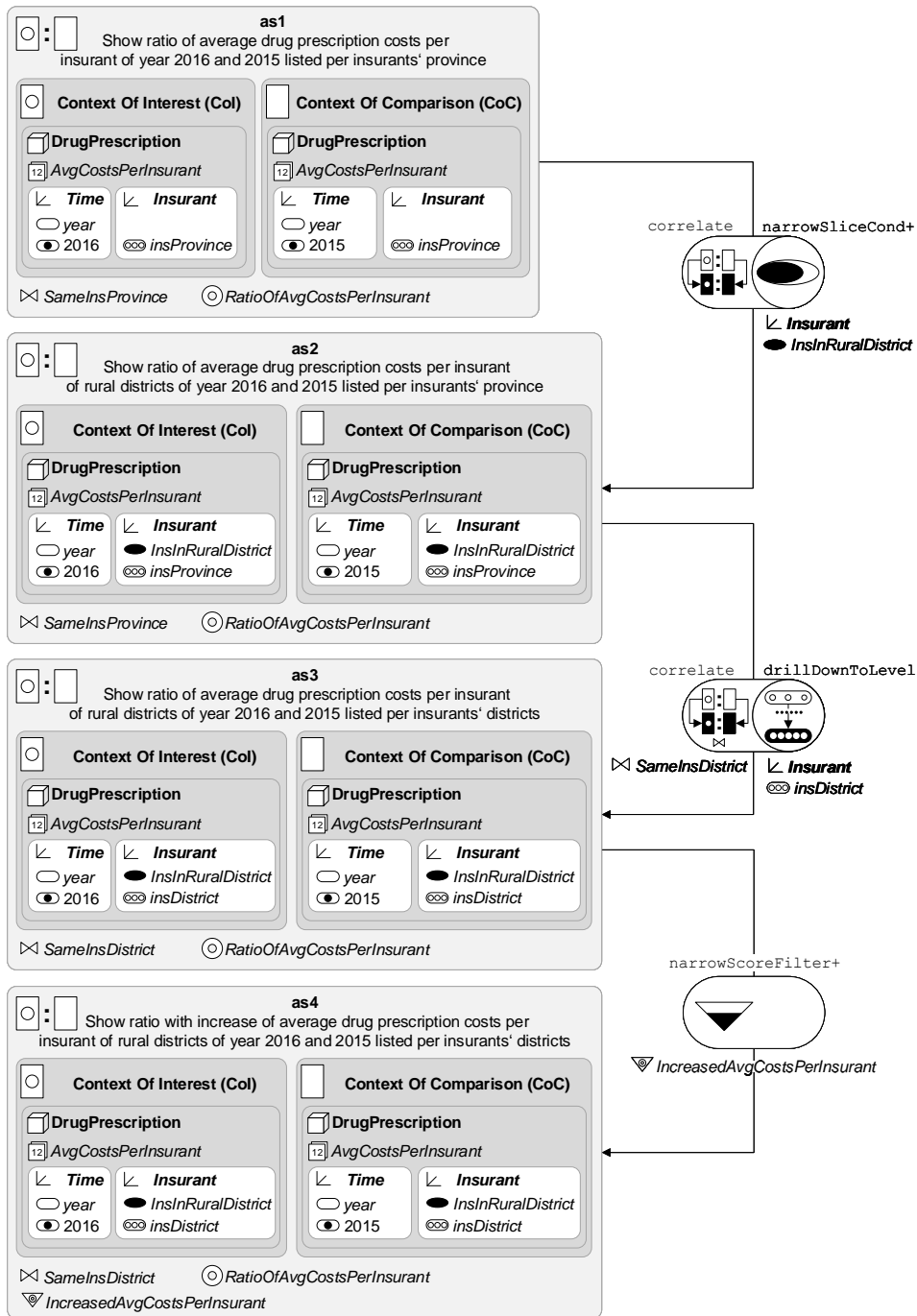


Figure 4.6: Example of three comparative navigation steps containing operators correlate and narrowScoreFilter+

by another score predicate in the target under the assumption that the new score predicate of the target implies the old one of the source. Again as a consequence, the overall score filter of the source subsumes the overall score filter of the target. Applying navigation operators `broadenScoreFilter-` or `broadenScoreFilter->` effects that the overall score filter of the comparative target analysis situation subsumes the overall score filter of the source. Whereas operator `broadenScoreFilter-` removes score predicates from the set of score filters, navigation operator `broadenScoreFilter->` replaces a single score predicate by another one. In the last case, the old score predicate must imply the new one to broaden the overall score filter of the target. Table 4.18 presents navigation operators `refocusScoreFilter` and `refocusScoreFilter->`. These operators can be used to change score filters such that no subsumption relation between source and target must be satisfied. Whereas operator `refocusScoreFilter` replaces the whole set of score filters, navigation operator `refocusScoreFilter->` only exchanges a single score predicate of the set of score filters.

The example of Figure 4.6 includes another navigation step that demonstrates operator `narrowScoreFilter+`. This navigation step links comparative source analysis situation `as3` to target `as4`. Score predicate *Increased-AvgCostsPerInsurant* defined in the eDFM of Figure 2.3 is added as a score filter to comparative target analysis situation `as4`. In this case, a result set is restricted only to result rows that comprise increased drug prescription costs concerning comparison of analysis situation `as4`. This comparative navigation step can be formally written as `as4 = as3.narrowScoreFilter+(Increased-AvgCostsPerInsurant)`.

The last comparative navigation operator of this section is specified in Table 4.19. Operator `rejoin` exists in two variants. In both variants, the set of join conditions is replaced. Additionally, in the first variant, one can also change scores and score filters. Navigation operator `rejoin` can be used, if there is a need to change the join conditions but not the granularity levels of dimension qualifications of the context of interest and the context of comparison. If there is also a requirement to modify granularities, one has to use operator `correlate` in combination with a non-comparative drill-down-

Table 4.17: Operators narrowScoreFilter+, narrowScoreFilter->, broadenScoreFilter-, and broadenScoreFilter->

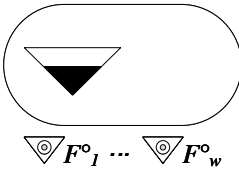
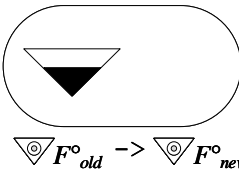
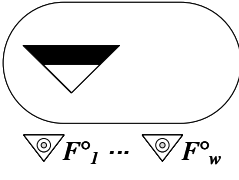
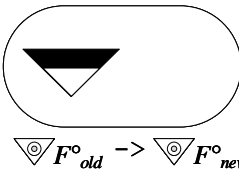
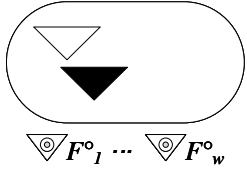
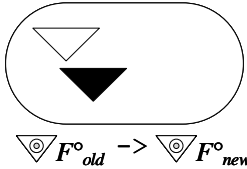
Operator Definition	Symbol
$\text{narrowScoreFilter}+(\{F_1^\circ, \dots, F_w^\circ\})$ <p>Precondition: $F_1^\circ, \dots, F_w^\circ$ are score predicates defined over $\text{Scores}_{\text{src}}$, $\text{AMsrs}_{\text{CoI}_{\text{src}}}$, and $\text{AMsrs}_{\text{CoC}_{\text{src}}}$, and $\{F_1^\circ, \dots, F_w^\circ\} \cap \text{ScoreFilters}_{\text{src}} = \emptyset$.</p> <p>Postcondition: $\text{ScoreFilters}_{\text{trg}} = \text{ScoreFilters}_{\text{src}} \cup \{F_1^\circ, \dots, F_w^\circ\}$</p>	<p>narrowScoreFilter+</p> 
$\text{narrowScoreFilter}\text{->}(F_{\text{old}}^\circ, F_{\text{new}}^\circ)$ <p>Precondition: $F_{\text{old}}^\circ \in \text{ScoreFilters}_{\text{src}}$, F_{new}° is a score predicate defined over $\text{Scores}_{\text{src}}$, $\text{AMsrs}_{\text{CoI}_{\text{src}}}$, and $\text{AMsrs}_{\text{CoC}_{\text{src}}}$, $F_{\text{new}}^\circ \notin \text{ScoreFilters}_{\text{src}}$, and $F_{\text{new}}^\circ \Rightarrow F_{\text{old}}^\circ$.</p> <p>Postcondition: $\text{ScoreFilters}_{\text{trg}} = \text{ScoreFilters}_{\text{src}} \setminus \{F_{\text{old}}^\circ\} \cup \{F_{\text{new}}^\circ\}$</p>	<p>narrowScoreFilter-></p> 
$\text{broadenScoreFilter}\text{-}(\{F_1^\circ, \dots, F_w^\circ\})$ <p>Precondition: $\{F_1^\circ, \dots, F_w^\circ\} \subseteq \text{ScoreFilters}_{\text{src}}$</p> <p>Postcondition: $\text{ScoreFilters}_{\text{trg}} = \text{ScoreFilters}_{\text{src}} \setminus \{F_1^\circ, \dots, F_w^\circ\}$</p>	<p>broadenScoreFilter-</p> 
$\text{broadenScoreFilter}\text{->}(F_{\text{old}}^\circ, F_{\text{new}}^\circ)$ <p>Precondition: $F_{\text{old}}^\circ \in \text{ScoreFilters}_{\text{src}}$, F_{new}° is a score predicate defined over $\text{Scores}_{\text{src}}$, $\text{AMsrs}_{\text{CoI}_{\text{src}}}$, and $\text{AMsrs}_{\text{CoC}_{\text{src}}}$, $F_{\text{new}}^\circ \notin \text{ScoreFilters}_{\text{src}}$, and $F_{\text{old}}^\circ \Rightarrow F_{\text{new}}^\circ$.</p> <p>Postcondition: $\text{ScoreFilters}_{\text{trg}} = \text{ScoreFilters}_{\text{src}} \setminus \{F_{\text{old}}^\circ\} \cup \{F_{\text{new}}^\circ\}$</p>	<p>broadenScoreFilter-></p> 

Table 4.18: Operators `refocusScoreFilter` and `refocusScoreFilter->`

Operator Definition	Symbol
$\text{refocusScoreFilter}(\{F_1^\circ, \dots, F_w^\circ\})$ Precondition: $F_1^\circ, \dots, F_w^\circ$ are score predicates defined over $Score_{src}$, $AMsrsCoI_{src}$, and $AMsrsCoC_{src}$, and $\{F_1^\circ, \dots, F_w^\circ\} \cap ScoreFilters_{src} = \emptyset$. Postcondition: $ScoreFilters_{trg} = \{F_1^\circ, \dots, F_w^\circ\}$	<code>refocusScoreFilter</code> 
$\text{refocusScoreFilter->}(F_{old}^\circ, F_{new}^\circ)$ Precondition: $F_{old}^\circ \in ScoreFilters_{src}$, F_{new}° is a score predicate defined over $Score_{src}$, $AMsrsCoI_{src}$, and $AMsrsCoC_{src}$, and $F_{new}^\circ \notin ScoreFilters_{src}$. Postcondition: $ScoreFilters_{trg} = ScoreFilters_{src} \setminus \{F_{old}^\circ\} \cup \{F_{new}^\circ\}$	<code>refocusScoreFilter-></code> 

operator.

4.3.3 Operators Dropping Comparison

After considering a comparative analysis situation, it can also be useful to focus the analysis process on either the context of interest or the context of comparison. In this case, starting from a comparative analysis situation, comparison is dropped and the user navigates to a non-comparative analysis situation.

In Table 4.20, two operators are specified that can be applied to realize such navigation steps. Whereas operator `unrelate` links a comparative analysis situation to a non-comparative one that is equal to the context of interest, navigation operator `untarget` has a non-comparative target analysis situation that corresponds to the context of comparison. In other words, operator `unrelate` drops comparison by dropping the context of comparison and operator `untarget` drops comparison by dropping the context of interest. Both navigation operators do not have parameters.

Figure 4.7 presents an example that demonstrates navigation operator

Table 4.19: Operator rejoin

Operator Definition	Symbol
$\text{rejoin}(\{J_1, \dots, J_t\}, \{S_1, \dots, S_v\}, \{F_1^\circ, \dots, F_w^\circ\})$ Precondition: J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $\text{DimSchemas}_{CoI_{src}}$ and $\text{DimSchemas}_{CoC_{src}}$; S_1, \dots, S_v are scores defined over aggregate measures in $\text{AMsrs}_{CoI_{src}}$ and $\text{AMsrs}_{CoC_{src}}$; $F_1^\circ, \dots, F_w^\circ$ are score predicates defined over $\{S_1, \dots, S_v\}$, $\text{AMsrs}_{CoI_{src}}$, and $\text{AMsrs}_{CoC_{src}}$. Postcondition: $\text{JoinConds}_{trg} = \{J_1, \dots, J_t\}$, $\text{Scores}_{trg} = \{S_1, \dots, S_v\}$, and $\text{ScoreFilters}_{trg} = \{F_1^\circ, \dots, F_w^\circ\}$.	<p style="text-align: center;">rejoin</p> <p style="text-align: center;">$J_1 \dots J_t$ $S_1 \dots S_v$ $F_1^\circ \dots F_w^\circ$</p>
$\text{rejoin}(\{J_1, \dots, J_t\})$ Precondition: J_1, \dots, J_t are join conditions defined over dimension levels of dimension schemas in $\text{DimSchemas}_{CoI_{src}}$ and $\text{DimSchemas}_{CoC_{src}}$. Postcondition: $\text{JoinConds}_{trg} = \{J_1, \dots, J_t\}$	<p style="text-align: center;">rejoin</p> <p style="text-align: center;">$J_1 \dots J_t$</p>

Table 4.20: Operators unrelate and untarget

Operator Definition	Symbol
$\text{unrelate}()$ Precondition: <i>No additional preconditions</i> Postcondition: $\text{trg} = CoI_{src}$	<p style="text-align: center;">unrelate</p>
$\text{untarget}()$ Precondition: <i>No additional preconditions</i> Postcondition: $\text{trg} = CoC_{src}$	<p style="text-align: center;">untarget</p>

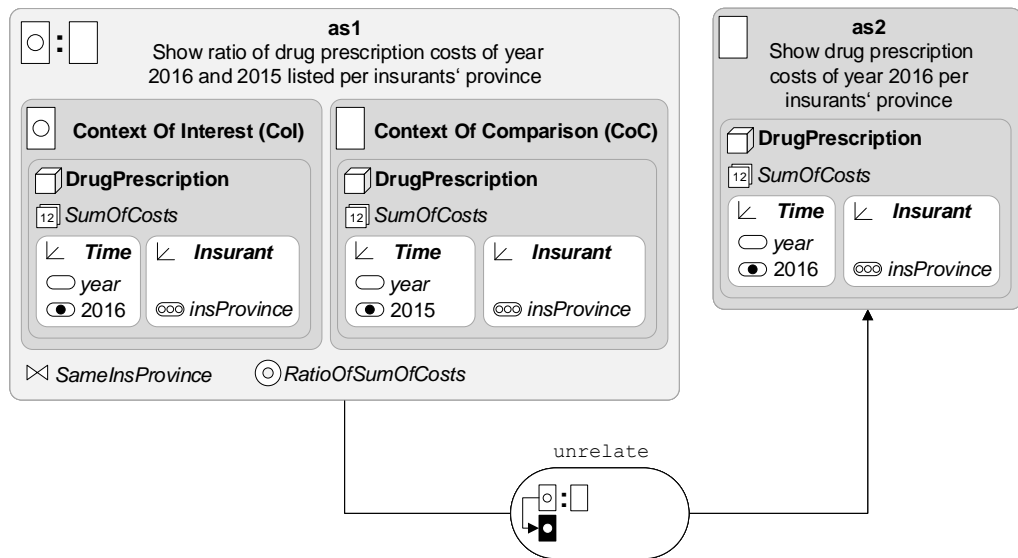


Figure 4.7: Example of a comparative navigation step containing operator `unrelate`

`unrelate`. In the comparative source analysis situation `as1`, drug prescription costs of year 2016 are compared with drug prescription costs of year 2015. The context of interest of analysis situation `as1` concerns the selection of drug prescription costs of year 2016, whereas, drug prescription costs of year 2015 are queried in the context of comparison. After comparison, the user wants to continue only to analyze drug prescription costs of year 2016. This can be attained by applying operator `unrelate` which returns non-comparative analysis situation `as2`, formally written as `as2 = as1.unrelate()`.

Operators `unrelate` and `untarget` were the last representatives that involve comparative analysis situations. In the next section, another navigation operator is introduced that does not only switch to another analysis situation but also introduces a new cube for further navigation.

4.4 Use of Non-Comparative Analysis Situations as Cubes

A non-comparative analysis situation is grounded on a cube but the result set of a non-comparative analysis situation can be also considered as a cube (derived cube). Such a derived cube can be used again by other non-comparative analysis situations and, of course, such a derived cube can also be used in comparative analysis situations as a context of interest or as a context of comparison. In this section, we present a navigation operator that uses the result of a non-comparative source analysis situation as a cube in a non-comparative target analysis situation. For defining such an operator, we introduce the notion of derived cube schemas and derived cube instances in the first subsection. Derived cubes do not have derived base measures, aggregate measures, aggregate measure predicates, scores, and score predicates. Thus, in the second subsection, such enrichments of cube schemas and cube instances are defined. Finally, in the last subsection, navigation operator `useAsCube` is presented.

4.4.1 Derived Cubes

In this subsection, we introduce dimension schemas, dimension instances, cube schemas, and cube instances that are derived from non-comparative analysis situations. Derived cube instances of derived cube schemas are generated by non-comparative analysis situations and can be used in other non-comparative analysis situations. First we start with the definition of a dimension schema which is derived from a dimension qualification.

Definition 4.4. Let DQ be a dimension qualification with $D = DimSchema_{DQ}$ and $G = GranLvl_{DQ} \neq top_D$. A dimension schema D' derived from DQ is constructed in the following way:

1. $Lvls_{D'} = \{G\} \cup \{L \in Lvls_D \mid G \twoheadrightarrow L \text{ with respect to } RollupRel_D\}$,
2. $RollupRel_{D'} = RollupRel_D \Big|_{Lvls_{D'}}$,⁷

⁷We use symbol $|$ for restriction of both functions and relations. Additionally, we claim

3. $DescrAttrs_{D'} = \{A \in DescrAttrs_D \mid LvlOfDescrAttr_D(A) \in LvlS_{D'}\}$,
4. $LvlOfDescrAttr_{D'} = LvlOfDescrAttr_D \Big|_{DescrAttrs_{D'}}$,
5. $DimOperators_{D'} = \{op \in DimOperators_D \mid LvlOfDimOperator_D(op) \in LvlS_{D'}\}$,
6. $LvlOfDimOperator_{D'} = LvlOfDimOperator_D \Big|_{DimOperators_{D'}}$,
7. $DimPredicates_{D'} = \{P \in DimPredicates_D \mid LvlOfDimPredicate_D(P) \in LvlS_{D'}\}$,
8. $LvlOfDimPredicate_{D'} = LvlOfDimPredicate_D \Big|_{DimPredicates_{D'}}$,
9. $JoinConditions_{D'} = \{P \in JoinConditions_D \mid LvlOfJoinCondition_D(P) \in LvlS_{D'}\}$, and
10. $LvlOfJoinCondition_{D'} = LvlOfJoinCondition_D \Big|_{JoinConditions_{D'}}$.

Moreover, in the context of dimension qualification DQ , we define $DerivedDimSchema_{DQ} = D'$ and for a non-comparative analysis situation \mathbf{as} , we define $DerivedDimSchemas_{\mathbf{as}} = \{D' \mid D' = DerivedDimSchema_{DQ}, DQ \in DimQuals_{\mathbf{as}}\}$. Furthermore, we define $DescrAttrsOfDerivedDimSchemas_{\mathbf{as}} = \bigcup_{D' \in DerivedDimSchemas_{\mathbf{as}}} DescrAttrs_{D'}$ for later use. \square

Definition 4.4 shows how a dimension schema can be derived from a dimension qualification that is a constituent of a non-comparative analysis situation. The derived dimension schema corresponds to the dimension schema of the dimension qualification except that the granularity level of the dimension qualification determines the base level of the derived dimension schema. Sublevels of the granularity level are omitted in the derived dimension schema. All other constituents of the dimension schema of the dimension qualification are transferred to the derived dimension schema provided that also the referred dimension level is transferred. Thus, roll-up relations, descriptive attributes, dimensional operators, dimensional predicates, and join

that this operator also restricts the range of a resulting function to its image.

conditions of the derived dimension schema are also parts of the dimension schema of the corresponding dimension qualification. Note that dimension schemas only can be derived from dimension qualifications, if they do not have the top level as granularity level.

In a derived cube schema of a non-comparative analysis situation, aggregate measures become simple base measures of the derived cube schema. To simplify definitions in this subsection, we define an auxiliary mapping that maps aggregate measures to simple base measures in a unique way.

Definition 4.5. *MapAMsrToSBMsr* is an injective mapping that maps an aggregate measure to a simple base measure. \square

The subsequent definition shows how a cube schema can be derived from a non-comparative analysis situation. This definition uses the notion of a derived dimension schema introduced previously.

Definition 4.6. Let **as** be a non-comparative analysis situation with $C = \text{CubeSchema}_{\mathbf{as}}$. A cube schema C' derived from **as** is constructed in the following way:

1. $\text{DimSchemas}_{C'} = \text{DerivedDimSchemas}_{\mathbf{as}}$,
2. $\text{BMsr}_{C'} = \text{MapAMsrToSBMsr}(\text{AMsr}_{\mathbf{as}}) \cup \{ A \mid A \in \text{DescrAttrsOf-DerivedDimSchemas}_{\mathbf{as}} \text{ and } A \text{ is numeric } \}$,
3. $\text{BMsrPredicates}_{C'} = \emptyset$,
4. $\text{AMsr}_{C'} = \emptyset$
5. $\text{AMsrPredicates}_{C'} = \emptyset$,
6. $\text{Scores}_{C'} = \emptyset$, and
7. $\text{ScorePredicates}_{C'} = \emptyset$.

Moreover, in the context of non-comparative analysis situation **as**, we define $\text{DerivedCubeSchema}_{\mathbf{as}} = C'$. \square

The non-comparative analysis situation provides all constituents of a derived cube schema. Dimension schemas are derived from its dimension qualifications. Aggregate measures of the non-comparative analysis situation become simple base measures and together with numeric descriptive attributes they are used as base measures of the derived cube schema. A derived cube schema contains no derived base measures, no base measure predicates, no aggregate measures, no aggregate measure predicates, no scores, and no score predicates—nevertheless, a derived cube schema also represents an eDFM. In the next subsection, we introduce the notion of an enrichment of a cube schema such that the target analysis situation of a navigation step containing operator `useAsCube` is not restricted to refer to a simple derived cube schema but can also refer to a cube schema with further enrichments as derived base measures, base measure predicates, aggregate measures, aggregate measure predicates, scores, or score predicates.⁸

In the following two definitions, derived cubes are considered at instance level. First we introduce the definition of a derived dimension instance followed by the definition of a derived cube instance that corresponds to the result set of the underlying non-comparative analysis situation.

Definition 4.7. Let \mathbf{as} be a non-comparative analysis situation, $\mathbf{c} = \text{CubeInstance}_{\mathbf{as}}$, $DQ \in \text{DimQuals}_{\mathbf{as}}$, $D = \text{DimSchema}_{DQ}$, $\mathbf{d} = \text{DimInstance}_{\mathbf{c}}(D)$, and $D' = \text{DerivedDimSchema}_{DQ}$. A *dimension instance* \mathbf{d}' of D' derived from \mathbf{d} (of dimension qualification DQ of non-comparative analysis situation \mathbf{as}) is constructed following:

1. $\text{Nodes}_{\mathbf{d}'} = \bigcup_{L \in \text{Lvls}_{D'}} \text{NodesOfLvl}_{\mathbf{d}}(L)$,
2. $\text{LvlOfNode}_{\mathbf{d}'} = \text{LvlOfNode}_{\mathbf{d}} \Big|_{\text{Nodes}_{\mathbf{d}'}}$,
3. $\text{NodeOrder}_{\mathbf{d}'} = \{ \prec \mid \prec \in \text{NodeOrder}_{\mathbf{d}} \text{ and } \prec \text{ is defined on } \text{NodesOfLvl}_{\mathbf{d}'}(L) \text{ for } L \in \text{Lvls}_{D'} \}$,
4. $\text{SuperNodeOf}_{\mathbf{d}'} = \{ f \in \text{SuperNodeOf}_{\mathbf{d}} \mid f : \text{NodesOfLvl}_{\mathbf{d}}(L_1) \rightarrow \text{NodesOfLvl}_{\mathbf{d}}(L_2) \text{ for } L_1, L_2 \in \text{Lvls}_{D'} \text{ and } L_1 \rightarrow L_2 \}$, and

⁸Especially, the definition of a non-comparative analysis situation claims to comprise at least one aggregate measure (see Definition 3.2).

5. $DescrAttrVals_{\mathbf{d}'} = \{f \in DescrAttrVals_{\mathbf{d}} \mid f \text{ has domain } NodesOfLvl_{\mathbf{d}}(L) \text{ for } L \in Lvl_{D'}\}$.

$DerivedDimInstances_{\mathbf{as}} = \{\mathbf{d}' \mid \mathbf{d}' \text{ is a dimension instance of dimension schema } D' \text{ derived from dimension instance } \mathbf{d} \text{ where } \mathbf{d} = DimInstance_{\mathbf{c}}(D), \mathbf{c} = CubeInstance_{\mathbf{as}}, D = DimSchema_{DQ}, D' = DerivedDimSchema_{DQ}, \text{ and } DQ \in DimQuals_{\mathbf{as}}\}$. \square

A derived dimension instance is grounded on the corresponding dimension instance of the cube instance used in the non-comparative analysis situation meaning that all dimension nodes of the derived dimension instance are also nodes of the dimension instance belonging to the non-comparative analysis situation. The dimension schema of the derived dimension instance is derived from the underlying dimension qualification. Each node of the derived dimension instance is mapped to the corresponding level of the derived dimension schema as it is done in the original dimension instance. Other properties as node orders, super-node relations, and values for descriptive attributes are also transferred from the original dimension instance. Using the definition of a derived dimension instance, we can introduce the definition of a derived cube instance.

Definition 4.8. Let \mathbf{as} be a non-comparative analysis situation, $\mathbf{c} = CubeInstance_{\mathbf{as}}$, and $C' = DerivedCubeSchema_{\mathbf{as}}$. A cube instance \mathbf{c}' of C' derived from \mathbf{c} (of non-comparative analysis situation \mathbf{as}) is constructed following:

1. $DimInstances_{\mathbf{c}'} = DerivedDimInstances_{\mathbf{as}}$ and
2. $Facts_{\mathbf{c}'} = ResultSet_{\mathbf{as}}$.

Moreover, in the context of non-comparative analysis situation \mathbf{as} , we define $DerivedCubeInstance_{\mathbf{as}} = \mathbf{c}'$. \square

Definition 4.8 shows how to construct a cube instance that is derived from the cube instance of a non-comparative analysis situation. The dimension instances of the derived cube instance correspond to the derived dimension instances and the facts correspond to the result set of the non-comparative analysis situation.

Note that dimension schemas, dimension instances, cube schemas, and cube instances are identified by names that can be obtained by function *NameOf*. Thus, especially derived dimension schemas, derived dimension instances, derived cube schemas, and derived cube instances also have names that can be received by function *NameOf*. For simplicity, one can use the name of the non-comparative analysis situation for both the name of the derived cube schema and the name of the derived cube instance.⁹ The names of derived dimension schemas and derived dimension instances can be the same as the names of the dimension schemas and dimension instances they are derived from.

With respect to SQL translation, derived cubes and derived dimensions are considered as SQL views. Again, one can use the name of the non-comparative analysis situation as a view name representing the corresponding derived cube instance. The definition of the view corresponds to the SQL translation of the non-comparative analysis situation. In this sense, a derived cube schema obtains semantics from the underlying non-comparative analysis situation.

4.4.2 Enrichments of Cubes

Derived cube schemas and derived cube instances do not contain (by definition) derived base measures, base measure predicate, aggregate measures, aggregate measure predicates, scores, and score predicates. To provide meaningful applications for navigation operator `useAsCube` introduced in the next subsection, we define the notion of enrichment of a cube schema and cube instance.¹⁰

Definition 4.9. Let C be a cube schema. A cube schema C' is an *enrichment* of cube schema C , if

⁹The definition of named analysis situation schemas and named analysis situations introduced in Chapter 6 explicitly comprise identifiers (names) for analysis situation schemas and analysis situations, respectively, that can also be used for naming derived cube schemas and instances of derived cube schemas.

¹⁰Note, of course, the notion of an enrichment can also be applied to derived cube schemas and derived cube instances as it is needed for navigation operator `useAsCube`.

1. $DimSchemas_{C'} = DimSchemas_C$,
2. $SimpleBMsrs_{C'} = SimpleBMsrs_C$,
3. $DerivedBMsrs_C \subseteq DerivedBMsrs_{C'}$,
4. $BMsrPredicates_C \subseteq BMsrPredicates_{C'}$,
5. $AMsrs_C \subseteq AMsrs_{C'}$
6. $AMsrPredicates_C \subseteq AMsrPredicates_{C'}$,
7. $Scores_C \subseteq Scores_{C'}$, and
8. $ScorePredicates_C \subseteq ScorePredicates_{C'}$. □

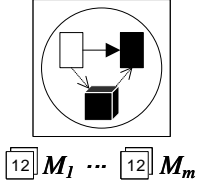
An enrichment of a cube schema can extend the set of derived base measures, the set of base measure predicates, the set of aggregate measures, the set of aggregate measure predicates, the set of scores, or the set of score predicates. In this way, we can introduce these sets for derived cube schemas. Based on enrichments of a cube schema, enrichments of a cube instance can be defined.

Definition 4.10. Let \mathbf{c} be a cube instance of cube schema C and let \mathbf{c}' be a cube instance of cube schema C' . Cube instance \mathbf{c}' is a *enrichment* of cube instance \mathbf{c} , if

1. cube schema C' is a *enrichment* of cube schema C ,
2. $DimInstances_{\mathbf{c}'} = DimInstances_{\mathbf{c}}$, and
3. $Facts_{\mathbf{c}'} = Facts_{\mathbf{c}}$. □

Note that a cube instance also comprises its cube schema. Thus, two cube instances \mathbf{c} and \mathbf{c}' can also be different, even if their dimension instances and facts are equal but provided that they have different cube schemas. In this sense, one can say that the notion of enrichment only refers the schema level of cubes.

Table 4.21: Operator `useAsCube`

Operator Definition	Symbol
<p>$\text{useAsCube}(\{M_1, \dots, M_m\})$</p> <p>Precondition: Source <code>src</code> is a non-comparative analysis situation and $\{M_1, \dots, M_m\} (\neq \emptyset) \subseteq AMsrs_C$ where C is an enrichment of $DerivedCubeSchema_{src}$.</p> <p>Postcondition: Target <code>trg</code> is a non-comparative analysis situation, $CubeInstance_{trg} = \mathbf{c}$ where $CubeSchema_{\mathbf{c}} = C$ and \mathbf{c} is an enrichment of $DerivedCubeInstance_{src}$, $BMsrConds_{trg} = \emptyset$, $AMsrs_{trg} = \{M_1, \dots, M_m\}$, $FilterConds_{trg} = \emptyset$, and, for $D \in DimSchemas_{\mathbf{c}}$, $DiceLvl_{trg}(D) = top$, $DiceNode_{trg}(D) = \mathbf{all}$, $SliceConds_{trg}(D) = \emptyset$, and $GranLvl_{trg}(D) = base$.</p>	<p style="text-align: center;">useAsCube</p> 

After defining the concepts of derived dimension schemas, derived dimension instances, derived cube schemas, derived cube instances, and enrichments of cube schemas and cube instances, in the next subsection, we continue to introduce navigation operator `useAsCube` that uses these concepts.

4.4.3 Operator Using Non-Comparative Analysis Situations as Cubes

In this subsection, we present navigation operator `useAsCube` which is specified in Table 4.21. It has a non-comparative analysis situation as source and another one as target. This operator generates a non-comparative target analysis situation and, additionally, it also provides a cube instance \mathbf{c} that is an enrichment of $DerivedCubeInstance_{src}$. Cube \mathbf{c} is a part of the definition of non-comparative target analysis situation `trg` and it is an instance of cube schema C which is an enrichment of $DerivedCubeSchema_{src}$. Because a non-comparative analysis situation also has to contain at least aggregate measures, operator `useAsCube` additionally introduces a non-empty set of aggregate measures $\{M_1, \dots, M_m\}$ that is a subset of $AMsrs_C$. Other components of target analysis situation `trg` are set to default values: The set

of base measure conditions and the set of filter conditions are empty sets, and, for each dimension schema of cube \mathbf{c} , the top level and the all node are used for dice level and dice node, the set of slice conditions is also an empty set, and the base level of the corresponding dimension schema is used as granularity level.

Figure 4.8 demonstrates an application of navigation operator `useAsCube`. Source analysis situation `DrugPrescrOfOldInsPerYearAndInsDistr` (subsequently also denoted by `src`) comprises cube `DrugPrescription`, aggregate measures `SumOfCosts`, `SumOfQuantity`, and `NumOfInsurants`, and three dimension qualifications concerning dimension schemas `Time`, `Insurant`, and `Drug`. Drug prescriptions of old insurants are selected and the sum of costs, the sum of quantity, and the number of insurants are calculated per year, insurants' district, and drug. The definitions of the aggregate measures and of dimensional predicate `OldInsurant` are presented in the eDFM of Figure 2.3. In Figure 4.9, select statement `Querysrc` is depicted that defines the semantics of non-comparative analysis situation `src`. Additionally, `Querysrc` is used to define SQL view `DrugPrescrOfOldInsPerYearAndInsDistr` that becomes the cube instance of non-comparative target analysis situation `AvgDrugPrescrOfOldInsPerYearAndInsDistr` (subsequently also denoted by `trg`). The name of source analysis situation `src` is also used as view name and as a name for `CubeInstancetrg`.¹¹

In Figure 4.11, the eDFM of Figure 2.3 was extended by cube schema `DrugPrescrOfOldInsPerYearAndInsDistr` (picked out by thick lines).¹² This cube schema is an enrichment of `DerivedCubeSchemasrc` of non-comparative source analysis situation `src` of Figure 4.8 and it also represents the cube schema for `CubeInstancetrg` (also denoted as `CubeSchematrg`). Again we also use the name of source analysis situation `src` as a name for `CubeSchematrg`.

Cube schema `DrugPrescrOfOldInsPerYearAndInsDistr` comprises simple

¹¹Later, in Chapter 6, we introduce named analysis situations used to define business intelligence (BI) analysis graphs. The name of a named analysis situation can be used as a view name.

¹²In the depiction of Figure 4.11, we do not present all the elements of the eDFM of Figure 2.3. Cube schemas `AmbTreatment` and `Hospitalization`, and dimension schemas `MedService` and `Hospital` are omitted due to lack of space.

base measures *quantity*, *costs*, and *numOfIns*. Within the red rectangle of cube schema *DrugPrescrOfOldInsPerYearAndInsDistr*, a new notation (with equal symbol) is used to indicate the mapping of aggregate measures to these three simple base measures. Aggregate measure *SumOfQuantity* is mapped to simple base measure *quantity*, *SumOfCosts* is mapped to *costs*, and aggregate measure *NumOfInsurants* is mapped to simple base measure *numOfIns*. This mapping is used to define cube schema *DerivedCubeSchema_{src}* which is enriched as depicted in Figure 4.11 leading to cube schema *DrugPrescrOfOldInsPerYearAndInsDistr* that is used as *CubeSchema_{trg}*.

Cube schema *DrugPrescrOfOldInsPerYearAndInsDistr* comprises dimension schemas *Drug*, *Time*, and *Insurant*. These are dimension schemas that are derived from the dimension qualifications of analysis situation **src**. For these derived dimension schemas, we use the same names as for dimension schemas *Drug*, *Time*, and *Insurant* contained in cube schema *DrugPrescription*. Nevertheless, two of them (*Time* and *Insurant*) are different in both cube schemas because, for cube schema *DrugPrescrOfOldInsPerYearAndInsDistr*, dimension schema *Time* only comprises level *year* (additionally to the top level) and dimension schema *Insurant* has dimension level *insDistrict* (and not level *insurant*) as base level. In Figure 4.11, the dashed lines lead to the base level of a dimension schema with respect to the corresponding cube schema. Note that only dimensional operators, dimensional predicates, and join conditions can be applied for cube schema *DrugPrescrOfOldInsPerYearAndInsDistr*, if they are defined on dimension levels that are also part of the corresponding derived dimension schema. Thus, for this cube schema, dimensional operators *monthOfPrevYear*, *monthOfNextYear*, *qrtOfPrevYear*, *qrtOfNextYear*, and dimensional predicate *OldInsurant* are not applicable.

Figure 4.11 also shows that cube schema *DrugPrescrOfOldInsPerYearAndInsDistr* contains derived base measure *costsPerUnit* and simple aggregate measures *SumOfQuantity*, *SumOfCosts*, and *NumOfInsurants*—all of them are based on simple base measures *quantity*, *costs*, and *numOfIns*.¹³

¹³Note that aggregate measures *SumOfQuantity*, *SumOfCosts*, and *NumOfInsurants* are used twice with respect to the definition of cube schema *DrugPrescrOfOldInsPerYearAndInsDistr*. First they are mapped to simple base measures which is necessary to obtain a derived cube schema from the non-comparative source analysis situation. Secondly, these

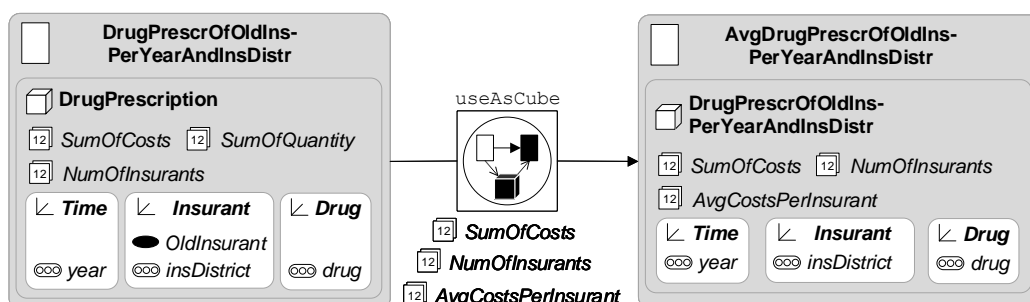


Figure 4.8: Example of a navigation step with operator `useAsCube`

In the case of simple aggregate measure *NumOfInsurants*, we introduce a new notation in Figure 4.11 that provides “alternative definitions”. For cube schema *DrugPrescription*, simple aggregate measure *NumOfInsurants* is defined by expression `COUNT(DISTINCT insurant)` that uses dimension level *insurant*. This dimension level cannot be used for the definition of *NumOfInsurants* in cube schema *DrugPrescrOfOldInsPerYearAndInsDistr*. Thus another defining expression is provided that uses simple base measure *numOfIns*: `SUM(numOfIns)`.¹⁴

Other components in Figure 4.11 that are “connected indirectly” to cube schema *DrugPrescrOfOldInsPerYearAndInsDistr* are also part of this cube schema. For instance, we can also use derived aggregate measure *AvgCostsPerInsurant*, aggregate measure predicate *HighAvgDrugPrescrCostsPerIns*, score *RatioOfAvgCostsPerInsurant*, join condition *SameInsProvince*, or dimensional predicate *InsInRuralDistrict*.

Navigation operator `useAsCube` in Figure 4.8 obtains a set of three aggregate measures as a parameter: *SumOfCosts*, *NumOfInsurants*, and *AvgCostsPerInsurant*. These aggregate measures become aggregate measures of non-comparative target analysis situation *AvgDrugPrescrOfOldInsPerYearAndInsDistr* (= **trg**).¹⁵ Target analysis situation **trg** comprises cube instance

aggregate measures are used in the non-comparative target analysis situation meaning that they are applied to cube instance *DrugPrescrOfOldInsPerYearAndInsDistr*.

¹⁴For this cube schema, we do not know the insurants themselves but we know the number of insurants per district. Thus we cannot count insurants but we can sum up the number of insurants per district.

¹⁵Formally, the navigation step can be written as for other navigation operators: **trg** =

```

CREATE OR REPLACE VIEW DrugPrescrOfOldInsPerYearAndInsDistr AS
SELECT  year, insDistrict, drug,
        SUM(costs) AS costs, SUM(quantity) AS quantity,
        COUNT(DISTINCT insurant) AS numOfIns
FROM    DrugPrescription NATURAL JOIN
        Time NATURAL JOIN
        Insurant NATURAL JOIN
        Drug
WHERE   insAge > 65
GROUP BY year, insDistrict, drug

```

Figure 4.9: SQL view of the derived cube of the non-comparative source analysis situation of the example in Figure 4.8

```

SELECT  year, insDistrict, drug,
        SUM(costs) AS SumOfCosts,
        SUM(numOfIns) AS NumOfInsurants,
        SUM(costs) / SUM(numOfIns) AS AvgCostsPerInsurant
FROM    DrugPrescrOfOldInsPerYearAndInsDistr NATURAL JOIN
        (SELECT DISTINCT year FROM Time) NATURAL JOIN
        (SELECT DISTINCT insDistrict, inhPerSqkmInInsDistr,
         insProvince FROM Insurant) NATURAL JOIN
        Drug
GROUP BY year, insDistrict, drug

```

Figure 4.10: SQL statement of the target analysis situation of the example in Figure 4.8

DrugPrescrOfOldInsPerYearAndInsDistr ($= CubeInstance_{trg}$)—an enrichment of *DerivedCubeInstance_{src}* corresponding to cube schema *DrugPrescrOfOldInsPerYearAndInsDistr* presented in Figure 4.11—a set of aggregate measures *AMsrs_{trg}* containing *SumOfCosts*, *NumOfInsurants*, and *AvgCostsPerInsurant*, and three dimension qualification with respect to dimension schemas *Time*, *Insurant*, and *Drug*. Note that dimension schemas *Time* and *Insurant* of cube schema *DrugPrescrOfOldInsPerYearAndInsDistr* have base level *year* and *insDistrict* which is a difference to dimension schemas *Time*

src.useAsCube ($\{SumOfCosts, NumOfInsurants, AvgCostsPerInsurant\}$).

and *Insurant* of cube schema *DrugPrescription* that is used in the source analysis situation. The dimension qualifications of the target analysis situation are set to the base level of the corresponding cube schema.

Whereas the view definition in Figure 4.9 demonstrates the SQL translation of the derived cube obtained from the source analysis situation, Figure 4.10 presents the SQL query of the target analysis situation ($Query_{trg}$). In the generation of $Query_{trg}$, two new specific requirements have to be respected: (1) Cube instance *DrugPrescrOfOldInsPerYearAndInsDistr* corresponds to a view definition and not to a fact table, and (2) there are no dimension tables with respect to the dimension qualifications corresponding to dimension schemas *Time* and *Insurant*. Whereas for the first requirement there is no need to adapt the query generation as introduced in Section 3.1.4, the second requirement involves some modifications. As one can see in Figure 4.10, instead joining dimension tables *Time* and *Insurant*,¹⁶ we define subselect statements that only provide those attributes defined in the dimension schemas of cube schema *DrugPrescrOfOldInsPerYearAndInsDistr*. The subselect statements use the original dimension tables *Time* and *Insurant*. To avoid multiple occurrence of same records, both subselects contain the *DISTINCT* keyword.

Navigation operator *useAsCube* was the last one presented in this chapter. It was a specific one because it does not only link a source to a target analysis situation but it also generates a new cube instance. The following last section of this chapter demonstrates navigation guards that can be used additionally in all kinds of navigation steps.

4.5 Navigation Steps Containing Navigation Guards

In Section 4.1 of this chapter, a generic definition of a navigation step (Definition 4.3) was presented that also comprises the notion of a navigation guard. Further remarks and demonstrations on navigation steps containing navigation guards will be given in the current section.

Figure 4.12 shows a graphical representation of an example of a comparative navigation step which is controlled by a navigation guard that examines whether the execution of query $Query_{as1}$ of comparative source analysis situation $as1$ has a non-empty result set. If $ResultSet_{as1} \neq \emptyset$, then $as1.hasResult()$ is true. Note that the boolean expression of the navigation guard is formally written as $as1.hasResult()$ expressing that operator $hasResult()$ is applied to analysis situation $as1$. In the graphical representation, a navigation guard is depicted as a rectangle (with a grey diamond in the upper left corner) containing expression $hasResult()$ (without analysis situation $as1$ as a prefix)—the context of analysis situation $as1$ is given implicitly by the arrow linkage.

Formally, the operator invocation is written as $as2 = as1.[hasResult()] correlate(drillDownToLevel(Insurant, insDistrict), SameInsDistrict)$. The navigation step is specified as $(as1.correlate(drillDownToLevel(Insurant, insDistrict), SameInsDistrict), as2, as1.hasResult())$. Note, accordingly to Definition 4.3, a navigation step contains a navigation guard as the last component of the triple as a fully qualified boolean expression $as1.hasResult()$.

Semantically, the navigation step in Figure 4.12 can be described in the following way: Comparative analysis situation $as1$ compares average drug prescription costs per insurant of year 2016 with those of previous year 2015. Comparison is performed per insurants' province and, additionally, the ratio

¹⁶We assume to have dimension tables that are part of a star schema and that correspond to the eDFM as presented in Figure 2.3 of Section 2.4. In this case, we have columns *year*, *quarter*, *month*, and *date* for dimension table *Time*, and *insProvince*, *insDistrict*, *inhPerSqkmInInsDistr*, *insurant*, and *insAge* for dimension table *Insurant*.

of the costs is computed as a score. Score filter *IncreasedAvgCostsPerInsurant* indicates that only those records are included in a result set, if this ratio represents a cost increase. For a specific execution of query $Query_{as1}$, the navigation guard examines whether there are provinces having cost increase. If there are such provinces, the navigation step drills down to insurants' districts yielding that in target *as2* the same comparison is performed per insurants' district. Note that in this case, all districts are compared and not only those that belong to a province with cost increase.¹⁷

After completion of the chapter about navigation operators, in the next chapter, we continue to introduce analysis situations and navigation steps at schema level. Navigation guards also appear at schema level where they obtain additional semantics in the sense that they can also control the generation of navigation steps and not only the execution of navigation steps.

¹⁷In Chapter 5 (Subsection 5.3.3, Figure 5.23), an example is presented where a drill down to districts is only performed for provinces with cost increase.

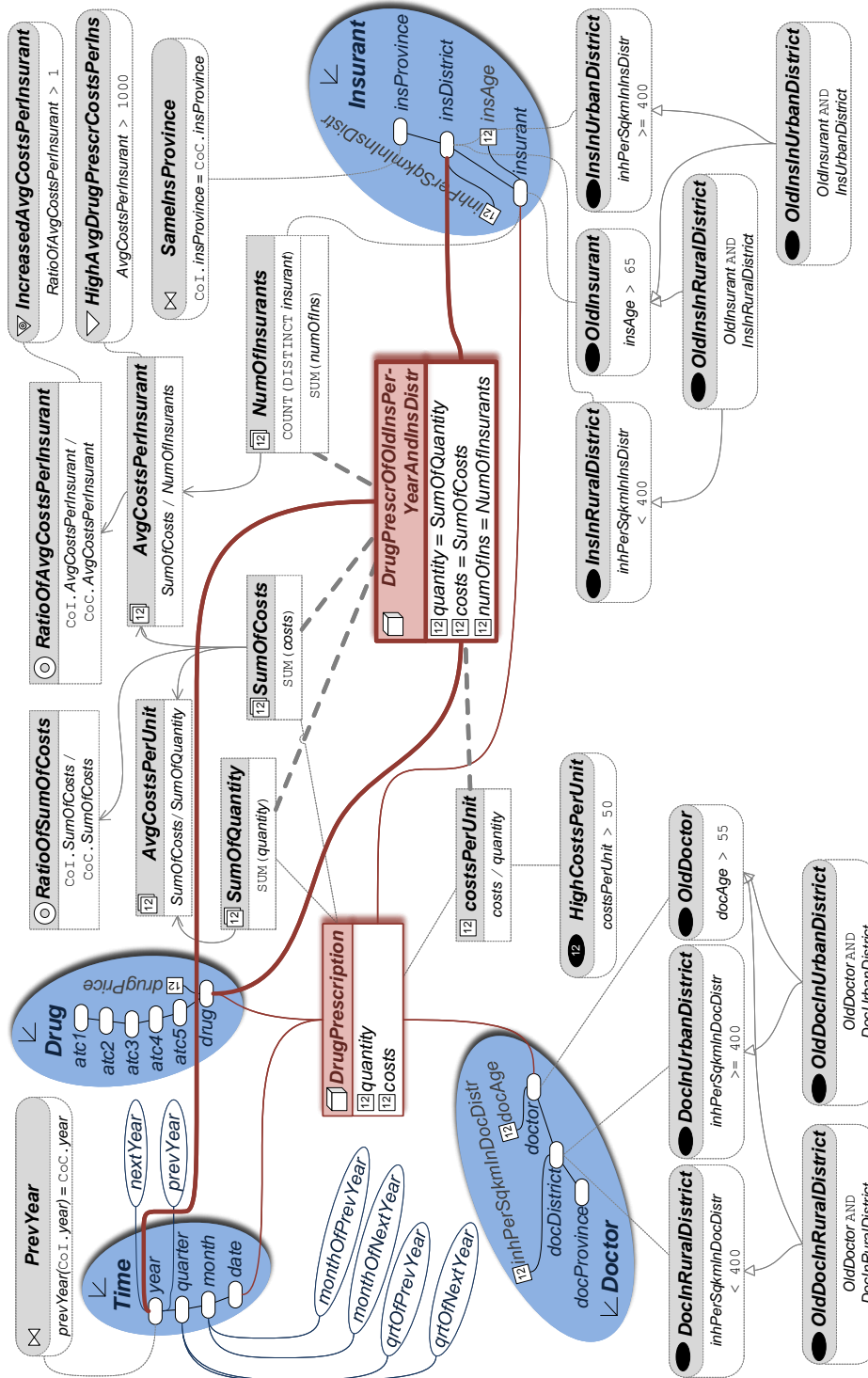


Figure 4.11: eDFM of the example in Figure 4.8

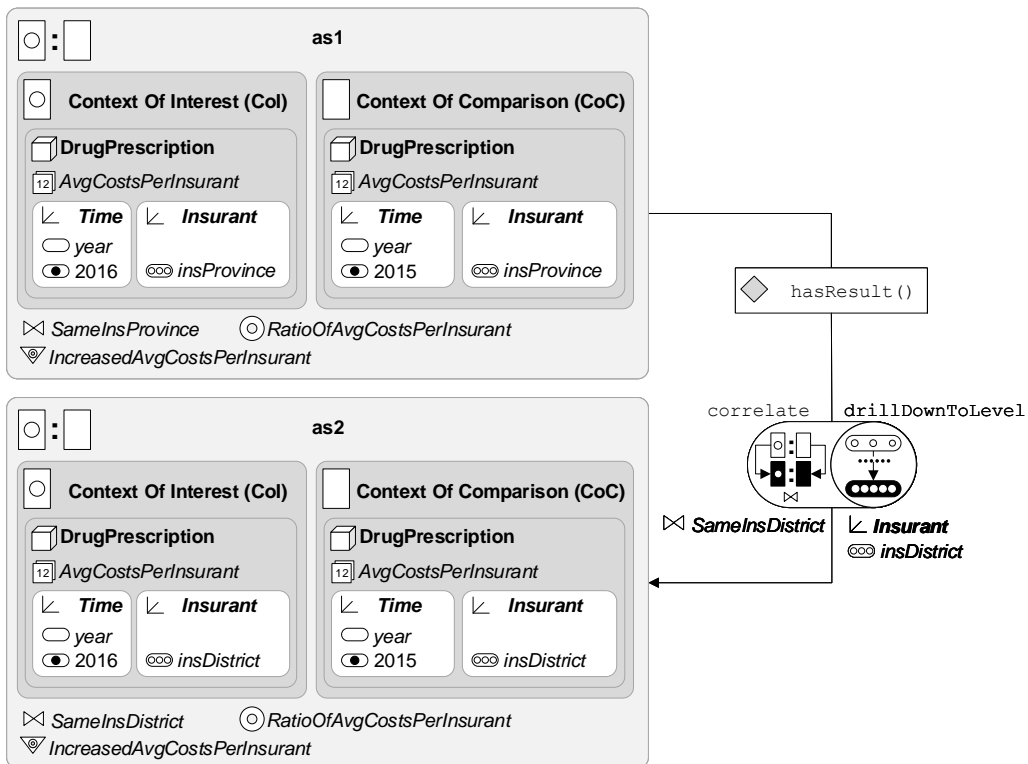


Figure 4.12: Example of a comparative navigation step containing a navigation guard

Chapter 5

Extension to Schema Level

Contents

5.1	Analysis Situation Schemas	222
5.1.1	Non-Comparative Analysis Situation Schemas . . .	222
5.1.2	Comparative Analysis Situation Schemas	230
5.1.3	Properties of Analysis Situation Schemas	234
5.2	Navigation Step Schemas	236
5.2.1	Generic Definitions	236
5.2.2	Operators Used in Navigation Guards	241
5.2.3	Type-Compliant Navigation Steps	245
5.3	Example of Navigation Step Schemas	250
5.3.1	Examples without Navigation Guards	252
5.3.2	Examples with Navigation Guards	264
5.3.3	Navigation Patterns	266
5.4	Discussion	274

In the previous chapters, we introduced analysis situations and navigation operators. An analysis situation represents a query that can be executed and which provides a result set. A navigation operator describes how a specific query can be transformed to receive another query. The navigation operator

describes the difference of both. With respect to WebML, a navigation operator can also be compared to contextual links of WebML. The link transfers content from a source to the target.

Up to now, analysis situations are restricted to specific queries that can be reused with no variation. To enhance re-usability, we introduce variables to all constituents of a non-comparative or comparative analysis situation. The cube schema an analysis situation is grounded on represents the only fixed prerequisite. Thus, for non-comparative analysis situations, we allow to use variables for the cube instance, the set of measures, the set of base measure conditions, the set of filter conditions, and we allow to use variables for the dice level, the dice node, the set of slice conditions, and the granularity level of a dimension qualification. As the context of interest and the context of comparison of a comparative analysis situation also represent non-comparative analysis situations, both contexts can contain variables, too. Additionally, in a comparative analysis situation, variables can be used for the set of join conditions, the set of scores, and the set of score filters.

In this sense, analysis situations and navigation steps are considered at schema level. We introduce the notion of analysis situation schemas and navigation step schemas. To obtain a specific query, one has to bind variables to concrete values. This can be interpreted as instantiation of analysis situation schemas and navigation step schemas. An analysis situation schema can be reused for several analysis situations and thus for several queries depending on the variable assignment. The only restrictions are given by the cube schema an analysis situation is grounded on and by the type of an analysis situation that indicates whether it is a non-comparative or comparative one.

The flexibility of an analysis situation schema depends on the use of variables and constants. There are two extreme cases: If one uses variables for all components, the resulting analysis situation schema can be used to instantiate all possible analysis situations with respect to the given cube schema. In the other case, if all components of an analysis situation schema are specified by constants, only one analysis situation can be instantiated. Thus, in this case, the analysis situation schema itself represents a specific

analysis situation instance (its single analysis situation instance).¹

To extend the concept of navigation steps to schema level, we also introduce navigation step schemas. Instead of analysis situations, a navigation step schema comprises analysis situation schemas as source and target. Moreover, it is allowed to use variables as actual parameters depending on the operator and the formal parameter. Similarly to analysis situation schemas, navigation step schemas can also be instantiated to obtain navigation steps where all variables are bound to constant values. A navigation step schema itself does not comprise an operator invocation but it contains the schema to create an operator invocation of the instantiated navigation step.

Additionally, we allow to specify navigation guards in navigation step schemas that are used to control navigation of instances of those navigation step schemas. A navigation guard at schema level represents a condition that is used to examine properties of the instantiated source analysis situation or to examine the result set of the executed query of the instantiated source analysis situation.² Only in the case that the navigation guard is true, one can navigate to the target analysis situation.

In this chapter, we first present analysis situation schemas (non-comparative and comparative), the notion of properties of an analysis situation and analysis situation schema (presented in a separated subsection), and, afterwards, we introduce navigation step schemas that also include navigation guards. The definition of navigation step schemas represents a generic one.

Additionally, in a separated subsection, we introduce the notion of type-compliant (i.e., schema-compliant) navigation steps which leads to a discussion about type checking. In this context, we distinguish between static and dynamic type checking, i.e., we discuss cases where type checking can be performed already at schema level and cases where type checking can be performed only at instance level.

Examples of navigation step schemas and exemplary navigation patterns are presented in a separate section of this chapter. The last section of this

¹This case can be compared with a singleton class in object-oriented design patterns.

²In the second case, a navigation guard corresponds to the navigation guard of navigation steps introduced in Chapter 4.

chapter concludes with an additional discussion referring to previous work.

5.1 Analysis Situation Schemas

This section presents non-comparative and comparative analysis situation schemas. An analysis situation schema can be considered as an analysis situation where some constituents are unbound. One can think of an analysis situation where each constituent represents a constant or an unbound variable (free variable). If each variable is bound, one obtains an instantiation of the analysis situation schema which corresponds to an analysis situation (analysis situation instance). An analysis situation schema can be instantiated arbitrary often, usually with different variable assignments. If all constituents of an analysis situation schema are constants, one obtains a specific case of an analysis situation schema that determines a single analysis situation instance (comparable with a singleton class in object-oriented design patterns). Thus, each analysis situation could also be considered as an analysis situation schema without free variables.

5.1.1 Non-Comparative Analysis Situation Schemas

The following definition introduces non-comparative analysis situation schemas. A non-comparative analysis situation schema is grounded on a cube schema. Each component of a non-comparative analysis situation schema can be specified by a constant or a variable.

Definition 5.1. A *non-comparative analysis situation schema* $AS = (C, \mathbf{c}, B, M, DQS, F)$ comprises the following constituents:

1. C is a cube schema,
2. \mathbf{c} is a constant cube instance of cube schema C or a variable that can be bound to a cube instance of cube schema C ,
3. B is a constant set of or a variable that can be bound to a set of base measure predicates comprising elements of $BMSrPredicates_C$,

4. M is a constant non-empty set of or a variable that can be bound to a non-empty set of aggregate measures comprising elements of $AMsrs_C$,
5. $DQS = \{DQ_1, \dots, DQ_n\}$ with $DQ_i = (D_i, L_i, N_i, P_i, G_i)$, for $1 \leq i \leq n$, $DimSchemas_C = \bigcup_{i=1}^n D_i$, and
 - (a) L_i is a constant element of or a variable that can be bound to an element of $Lvls_{D_i}$,
 - (b) N_i is a constant dimension node of or a variable that can be bound to a dimension node of a dimension instance \mathbf{d} of D_i that belongs to cube instance \mathbf{c} with $LvlOfNode_{\mathbf{d}}(N_i) = L_i$ (additionally, we require that, if L_i is a variable, then N_i also is a variable),
 - (c) P_i is a constant set of or a variable that can be bound to a set of dimensional predicates comprising elements of $DimPredicates_{D_i}$,
 - (d) G_i is a constant element of or a variable that can be bound to an element of $Lvls_{D_i}$, and
6. F is a constant set of or a variable that can be bound to a set of aggregate measure predicates comprising elements of $AMsrPredicates_C$.

Moreover, we define $CubeSchema_{AS} = C$, $CubeInstance_{AS} = \mathbf{c}$, $BMSrConds_{AS} = B$, $AMsrs_{AS} = M$, $FilterConds_{AS} = F$, $DimQuals_{AS} = DQS$, and $DimSchemas_{AS} = DimSchemas_C$. For $DQ \in DQS$ with $DQ = (D, L, N, P, G)$, we define $DimSchema_{DQ} = D$, $DimQual_{AS}(D) = DQ$, $DiceLvl_{AS}(D) = L$, $DiceNode_{AS}(D) = N$, $SliceConds_{AS}(D) = P$, and $GranLvl_{AS}(D) = G$.

Furthermore, we allow to use symbol $?$ to denote a variable (without introducing a variable name). Thus in the case of variables, one can write $CubeInstance_{AS} = ?$, $BMSrConds_{AS} = ?$, $AMsrs_{AS} = ?$, $FilterConds_{AS} = ?$, and, for $D \in DimSchemas_{AS}$, $DiceLvl_{AS}(D) = ?$, $DiceNode_{AS}(D) = ?$, $SliceConds_{AS}(D) = ?$, and $GranLvl_{AS}(D) = ?$. \square

We allow to use variables for the cube instance, for the set of aggregate measures, for the set of base measure conditions, and for the set of filter conditions of a non-comparative analysis situation schema, and for the

constituents of a dimension qualification, i.e., we allow to use variables for the dice level, for the dice node, for the set of slice conditions, and for the granularity level of a dimension qualification with respect to a dimension schema. Note that it is not allowed to use variables for the cube schema of a non-comparative analysis situation schema and for denoting the dimension schema of a dimension qualification. As it is also possible to use variables for cube instances, it is also required to include the underlying cube schema to the definition of a non-comparative analysis situation schema.³ Furthermore, for aggregate measures, base measure conditions, filter conditions, and slice conditions, it is allowed to use variables for the whole set but not for single elements of such sets.

An element of a non-comparative analysis situation schema (for instance, the dice node $DiceNode_{AS}(D)$ of the dimension qualification with respect to dimension schema D of analysis situation schema AS) could be a constant value or an unbound variable. In the case of an unbound variable, one can use symbol $?$: $DiceNode_{AS}(D) = ?$. Additionally, we allow to append a name to this variable symbol. This adds no further formal semantics but only increases readability. As an example, if the variable for dice nodes should represent dimension nodes of level year, one can denote this variable as $?year$ instead by only using symbol $?$: $DiceNode_{AS}(D) = ?year$.

In graphical representations, analysis situation schemas are drawn like analysis situations but with a double-edged boundary (instead of a single-edged boundary). In Figure 5.1, an example of a non-comparative analysis situation schema with unbound variables is depicted in full graphical representation. The name of an analysis situation schema (in our example *Drug-PrescrCosts*) is written in a *slant* font. Variables are also denoted in the graphical representation by symbol $?$.⁴ The example contains variables for the set of base measure conditions and for the set of filter conditions. In the dimension qualification of dimension schema *Time*, there are variables

³As a difference, the definition of a non-comparative analysis situation does not contain the cube schema as an explicit component because the given cube instance itself comprises the cube schema implicitly.

⁴Although it would be also possible in the graphical representation, this example does not use named variables as, for instance, $?year$.

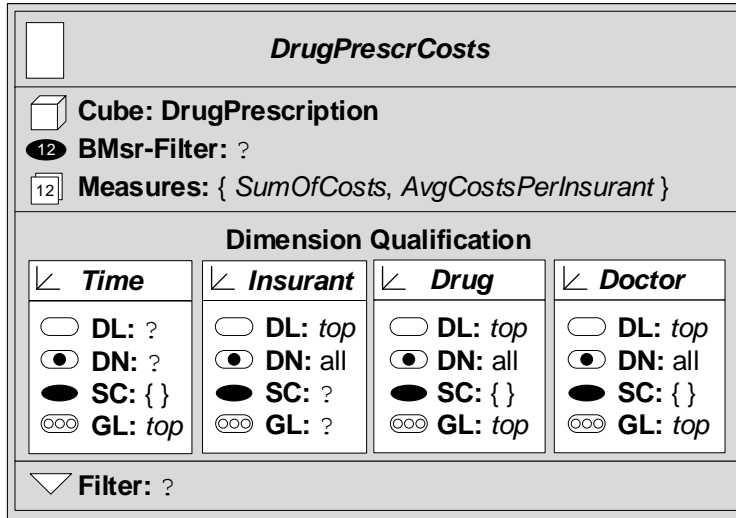


Figure 5.1: Non-comparative analysis situation schema with unbound variables (full graphical representation)

for the dice level and the dice node, and, in the dimension qualification of dimension schema *Insurant*, there are variables for the set of slice conditions and the granularity level. Note, in the case of sets, there is one variable for the whole set but not for a single set element which is not supported by our definition of analysis situation schemas.

Figure 5.2 presents the same example as shown in Figure 5.1 in a lean graphical presentation. Note again, that for set variables, symbol ? represents the whole set and not a single set element. This is important to emphasized because in the lean graphical representation, set elements are depicted in a loose notation without set braces. For instance, in Figure 5.2, aggregate measures *SumOfCosts* and *AvgCostsPerInsurant* represent two elements of the set of aggregate measures of non-comparative analysis situation schema *DrugPrescrCosts* whereas the depicted variable for base measure conditions represents the whole set of base measure conditions and not a single set element.

In Figure 5.3, the same example of Figure 5.1 and Figure 5.2 is depicted in a condensed representation. As for analysis situations, this representation offers no information to derive a precise definition of the analysis situation

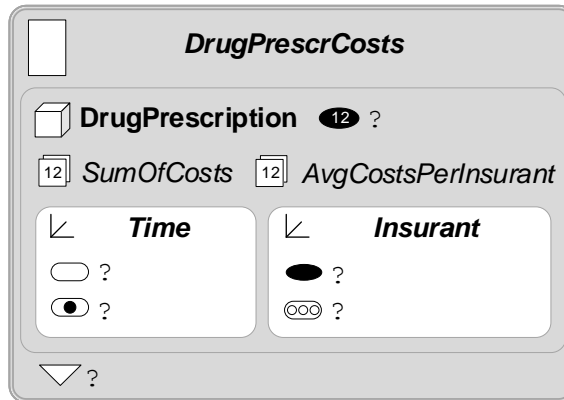


Figure 5.2: Non-comparative analysis situation schema with unbound variables (lean graphical representation)



Figure 5.3: Non-comparative analysis situation schema with unbound variables (condensed graphical representation)

schema. To distinguish between analysis situations and analysis situation schemas, we also use double-edged boundary for drawing analysis situation schemas in condensed graphical representation.

Figure 5.4 and Figure 5.5 point out two extreme cases. In Figure 5.4, there are no variables. All components of the non-comparative analysis situation schema are bound. This analysis situation schema exactly describes the analysis situation of Figure 3.4. In Figure 5.5, there are no constants. All components of this non-comparative analysis situation schema are free variables that must be bound at instantiation time. The underlying cube schema and its dimension schemas are the only information that is required as a fixed prerequisite. Note that in this analysis situation schema, the cube instance is also represented by an unbound variable. But in this case, we do not use symbol ? in the graphical representation. Instead we indicate the

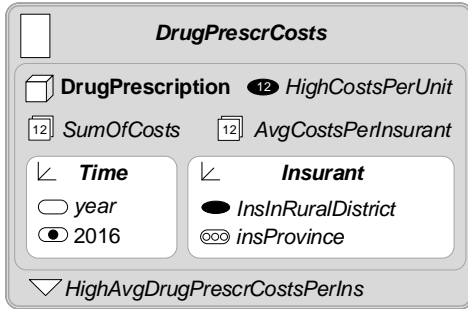


Figure 5.4: Non-comparative analysis situation schema without variables (lean graphical representation)

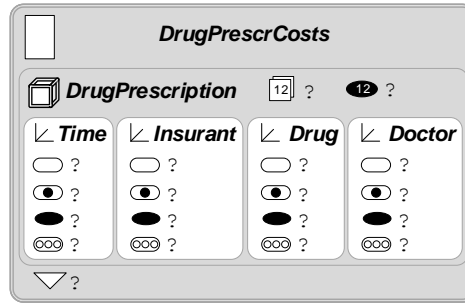


Figure 5.5: Non-comparative analysis situation schema without constants (lean graphical representation)

underlying cube schema prefixed by an double-edged cube symbol.⁵ By this convention, a variable for a cube instance is specified implicitly in a graphical representation.

Figure 5.6 is similar to the example in Figure 5.5 except that the dice node in dimension schema *Time* represents a named variable *?year*. To obtain a meaningful example, the dice level of dimension schema *Time* is restricted to dimension level *year*. Note that variable name *?year* is only used for better readability but does not introduce additional formal semantics.

The following definition specifies an instance of a non-comparative analysis situation schema where all variables are bound. An instance of a non-comparative analysis situation schema represents a non-comparative analysis situation as defined in Section 3.1.

Definition 5.2. An *instance* $as = (CubeInstance_{as}, BMrConds_{as}, AMsr_{as}, DimQuals_{as}, FilterConds_{as})$ of a non-comparative analysis situation schema $AS = (C, \mathbf{c}, B, M, DQS, F)$ is a non-comparative analysis situation such that

1. $CubeInstance_{as} = \bar{\mathbf{c}}$, where $\bar{\mathbf{c}} = \mathbf{c}$, if \mathbf{c} is a constant, or, in the case that \mathbf{c} is a variable, $\bar{\mathbf{c}}$ is a constant to which variable \mathbf{c} is bound,

⁵Additionally, the name of the cube schema is written in a *slant* font whereas, as a difference, the name of a cube instance is given by a *sans serif* font. Note, that in our examples, the presented cube schemas and cube instances have the same name.

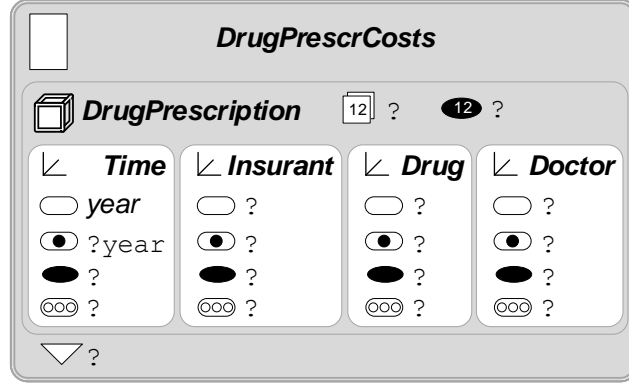


Figure 5.6: Non-comparative analysis situation schema with named variables (lean graphical representation)

2. $BMsrs_{as} = \bar{B}$, where $\bar{B} = B$, if B is a constant set, or, in the case that B is a set variable, \bar{B} is a constant set to which set variable B is bound,
3. $AMsrs_{as} = \bar{M}$, where $\bar{M} = M$, if M is a constant set, or, in the case that M is a set variable, \bar{M} is a constant set to which set variable M is bound,
4. for $(D, L, N, P, G) \in DQS$, there is exactly one dimension qualification $DQ \in DimQuals_{as}$ such that
 - (a) $DimSchema_{DQ} = D$,
 - (b) $DiceLvl_{DQ} = \bar{L}$, where $\bar{L} = L$, if L is a constant, or, in the case that L is a variable, \bar{L} is a constant to which variable L is bound,
 - (c) $DiceNode_{DQ} = \bar{N}$, where $\bar{N} = N$, if N is a constant, or, in the case that N is a variable, \bar{N} is a constant to which variable N is bound,
 - (d) $SliceConds_{DQ} = \bar{P}$, where $\bar{P} = P$, if P is a constant set, or, in the case that P is a set variable, \bar{P} is a constant set to which set variable P is bound,

- (e) $GranLvl_{DQ} = \bar{G}$, where $\bar{G} = G$, if G is a constant, or, in the case that G is a variable, \bar{G} is a constant to which variable G is bound, and
5. $FilterConds_{as} = \bar{F}$, where $\bar{F} = F$, if F is a constant set, or, in the case that F is a set variable, \bar{F} is a constant set to which set variable F is bound. \square

The example of a non-comparative analysis situation in Figure 3.4 represents an instance of the non-comparative analysis situation schema depicted in Figure 5.2. Set $\{HighCostsPerUnit\}$ is used to bind the set variable for base measure conditions. In the dimension qualification of dimension schema *Time*, there are variables for dice level and dice node that are bound to dimension level *year* and dimension node *2016*. The variables for the set of slice conditions and for the granularity level both concerning the dimension qualification of dimension schema *Insurant* are bound to set $\{InsInRuralDistrict\}$ and to dimension level *insProvince*. Finally, set $\{HighAvgDrugPrescrCostsPerIns\}$ is used to bind the variable for the set of filter conditions. Note that, if one uses other values for variable bindings, for example dimension node *2017* for the dice node with respect to dimension schema *Time* and the empty set for the set of filter conditions, one obtains other instances of this non-comparative analysis situation schema.

Furthermore, the example in Figure 3.4 could also be considered as an instance of the non-comparative analysis situation schemas depicted in Figures 5.4, 5.5, and 5.6. Whereas in Figure 5.4, the analysis situation schema does not comprise free variables and has the same constants as the analysis situation of Figure 3.4, Figure 5.5 contains free variables wherever possible. Additionally to the analysis situation schema in Figure 5.2, Figure 5.5 also comprises free variables for the cube instance, for the set of aggregate measures, and for each component of each dimension qualification. The analysis situation schema in Figure 5.6 is similar to the one of Figure 5.5, except that with respect to dimension schema *Time*, there exists a constant dice level *year* and a named dice node variable *?year*. But also in this case, the example in Figure 3.4 represents an instance of the non-comparative analysis

situation schema in Figure 5.6.

5.1.2 Comparative Analysis Situation Schemas

Analogously to non-comparative analysis situation schemas, one can specify and use schemas for comparative analysis situations. Comparative analysis situation schemas comprise two non-comparative analysis situation schemas for context of interest and context of comparison. Both can contain unbound variables. Additionally, we allow unbound variables for the set of join conditions, for the set of scores, and for the set of score filters. The following definition summarizes these ideas of comparative analysis situation schemas.

Definition 5.3. A *comparative analysis situation schema* $CAS = (AS^I, AS^C, J, S, SF)$ comprises the following constituents:

1. AS^I is a non-comparative analysis situation schema (for the *context of interest*, abbreviated as *CoI*),
2. AS^C is a non-comparative analysis situation schema with $CubeSchema_{AS^I} = CubeSchema_{AS^C}$ (for the *context of comparison*, abbreviated as *CoC*),
3. J is a constant set of or a variable that can be bound to a set of join conditions which are defined over qualified dimension levels of dimension schemas in $DimSchemas_{AS^I}$ ($= DimSchemas_{AS^C}$) qualified by **CoI** and **CoC** such that dimension levels qualified by **CoI** refer to AS^I and dimension levels qualified by **CoC** refer to AS^C ,
4. S is a constant set of or a variable that can be bound to a set of scores defined over qualified aggregate measures in $AMsrs_{AS^I}$ and $AMsrs_{AS^C}$ qualified by **CoI** and **CoC** such that aggregate measures qualified by **CoI** refer to AS^I and aggregate measures qualified by **CoC** refer to AS^C , and
5. SF is a constant set of or a variable that can be bound to a set of score predicates which are defined over S , $AMsrs_{AS^I}$, and $AMsrs_{AS^C}$.

CompDrugPrescrCosts							
Context of Interest (CoI)				Context of Comparison (CoC)			
Cube: DrugPrescription BMSr-Filter: ? Measures: { SumOfCosts, AvgCostsPerInsurant }				Cube: DrugPrescription BMSr-Filter: ? Measures: { SumOfCosts, AvgCostsPerInsurant }			
Dimension Qualification				Dimension Qualification			
Time	Insurant	Drug	Doctor	Time	Insurant	Drug	Doctor
DL: ? DN: ? SC: { } GL: top	DL: top DN: all SC: ? GL: ?	DL: top DN: all SC: { } GL: top	DL: top DN: all SC: { } GL: top	DL: ? DN: ? SC: { } GL: top	DL: top DN: all SC: ? GL: ?	DL: top DN: all SC: { } GL: top	DL: top DN: all SC: { } GL: top
Filter: ?				Filter: { }			
Join Condition: { SameInsProvince }							
Scores: { RatioOfSumOfCosts, RatioOfAvgCostsPerInsurant }							
Score Filter: ?							

Figure 5.7: Comparative analysis situation schema with unbound variables (full graphical representation)

Moreover, we define $CubeSchema_{CAS} = CubeSchema_{AS^I} (= CubeSchema_{AS^C})$, $CoI_{CAS} = AS^I$, $CoC_{CAS} = AS^C$, $JoinConds_{CAS} = J$, $Scores_{CAS} = S$, and $ScoreFilters_{CAS} = SF$.

Furthermore, we allow to use symbol ? to denote a variable (without introducing a variable name). Thus in the case of variables, one can write $JoinConds_{CAS} = ?$, $Scores_{CAS} = ?$, and $ScoreFilters_{CAS} = ?$. \square

Figure 5.7 shows an example of a comparative analysis situation schema with unbound variables in a full graphical representation. In Figure 5.8, the same example as in Figure 5.7 is depicted in lean graphical representation.⁶ Again comparative analysis situation schemas are drawn with double-edged boundaries. The context of interest and the context of comparison have almost the same non-comparative analysis situation schemas, except for the set of filter conditions where there is an unbound variable in the context of interest and an empty set as a constant value in the context of comparison.

⁶Finally, comparative analysis situation schemas also can be depicted in condensed graphical notation (not depicted here but used in figures of subsequent sections).

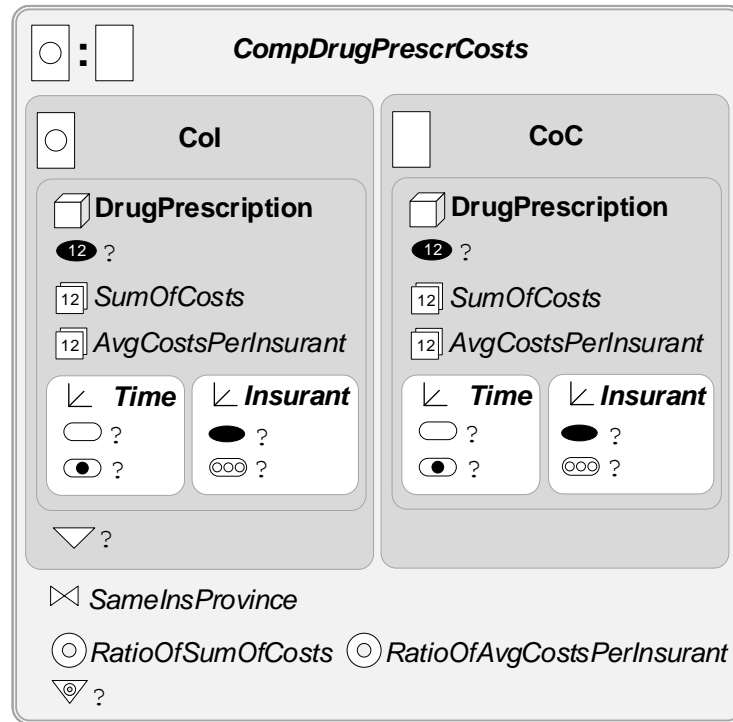


Figure 5.8: Comparative analysis situation schema with unbound variables (lean graphical notation)

Both contexts contain variables for the set of base measure conditions, for the dice level and the dice node with respect to dimension schema *Time*, and for the set of slice conditions and the granularity level with respect to dimension schema *Insurant*. But note that unbound variables of both contexts can be bound to different values at instantiation time. As an example concerning dimension schema *Time*, for the context of interest, the unbound variables for dice level and dice node can be bound to values *year* and *2016*, and values *year* and *2015* can be assigned to unbound variables for dice level and dice node of the context of comparison. If one wants to express that the dice node of the context of interest contains a specific year and the dice node of the context of comparison contains the previous year, one could use names for unbound variables (for example, *?year* and *?prevYear*). But note, this does not influence semantics—these are merely names. The semantics

that $?prevYear$ is the preceding year of $?year$ needs to be represented by a navigation operation (see next Section 5.2).

Finally, we emphasize that it is also allowed to use variables for cube instances. Of course, these cube instances have to be of the same cube schema. In the case of variable cube instances, we use the same graphical conventions as for non-comparative analysis situations schemas meaning that we do not use symbol $?$ to denote the variable but we insert the cube schema name prefixed by a double-edged cube pictogram.

Now we present the definition of an instance of a comparative analysis situation schema which corresponds to a comparative analysis situation. Note that such an instance includes two instances of a non-comparative analysis situation schema concerning the context of interest and the context of comparison.

Definition 5.4. An *instance* $cas = (CoI_{cas}, CoC_{cas}, JoinConds_{cas}, Scores_{cas}, ScoreFilters_{cas})$ of a comparative analysis situation schema $CAS = (AS^I, AS^C, J, S, SF)$ is a comparative analysis situation such that

1. CoI_{cas} is an instance of CoI_{CAS} ,
2. CoC_{cas} is an instance of CoC_{CAS} ,
3. $JoinConds_{cas} = \bar{J}$, where $\bar{J} = J$, if J is a constant set, or, in the case that J is a set variable, \bar{J} is a constant set to which set variable J is bound,
4. $Scores_{cas} = \bar{S}$, where $\bar{S} = S$, if S is a constant set, or, in the case that S is a set variable, \bar{S} is a constant set to which set variable S is bound, and
5. $ScoreFilters_{cas} = \overline{SF}$, where $\overline{SF} = SF$, if SF is a constant set, or, in the case that SF is a set variable, \overline{SF} is a constant set to which set variable SF is bound. □

The comparative analysis situation presented in Figure 3.12 and Figure 3.14 of Section 3.2 represents an instance of the comparative analysis situation schema depicted in Figure 5.7 and Figure 5.8. In this example, the

variables for the base measure conditions for both the context of interest and the context of comparison are bound to the set $\{HighCostsPerUnit\}$. For both contexts, the dice level variables with respect to dimension schema *Time* are bound to dimension level *year* and, with respect to dimension schema *Insurant*, the variables for the set of slice conditions are bound to set $\{InsInRuralDistrict\}$ and the variables for the granularity level are bound to dimension level *insProvince*. Dimension nodes 2016 and 2015 are set to the variables for the dice nodes of dimension schema *Time* for both the context of interest and the context of comparison. The variable for the set of filter conditions in the context of interest is bound to set $\{HighDrugPrescrCostsPerIns\}$ and the variable for the set of score filters is bound to set $\{IncreasedAvgCostsPerInsurant\}$.

5.1.3 Properties of Analysis Situation Schemas

In the definition of this subsection, we introduce the technical term *property* (or, alternatively, *item* or *component* as synonyms for *property*) of a dimension qualification, of an analysis situation, and of an analysis situation schema. The notions *property*, *item*, and *component* are used in the present chapter, especially for specifying navigation guards at schema level and for the discussion about type compliance.

Definition 5.5. The technical term *property* (or, alternatively, *item* or *component*) of a dimension qualification, an analysis situation, and an analysis situation schema is defined by the following enumeration:

1. The dimension schema, the dice level, the dice node, the set of slice conditions, and the granularity level of a dimension qualification are called *properties* (or, alternatively, *items* or *components*) of a dimension qualification.
2. The cube instance, the set of base measure predicates, the set of aggregate measures, the set of aggregate measure predicates, and the properties of all dimension qualifications of a non-comparative analysis

situation are called *properties* (or, alternatively, *items* or *components*) of a *non-comparative analysis situation*.

3. The set of join conditions, the set of scores, the set of score predicates, the properties of the context of interest, and the properties of the context of comparison of a comparative analysis situation are called *properties* (or, alternatively, *items* or *components*) of a *comparative analysis situation*.
4. The cube schema, the cube instance, the set of base measure predicates, the set of aggregate measures, the set of aggregate measure predicates, and the properties of all dimension qualifications of a non-comparative analysis situation schema are called *properties* (or, alternatively, *items* or *components*) of a *non-comparative analysis situation schema*.
5. The cube schema, the set of join conditions, the set of scores, the set of score predicates, the properties of the context of interest, and the properties of the context of comparison of a comparative analysis situation schema are called *properties* (or, alternatively, *items* or *components*) of a *comparative analysis situation schema*. \square

Note that at schema level, for almost all properties, it is allowed to be either a constant or a variable. Only dimension schemas and cube schemas always have to be constants, i.e., variables are not allowed for these items (see Definition 5.1).

In the context of navigation guards, for example, we use the introduced notions of this subsection (property, item, component). Table 5.1 (Subsection 5.2.2) contains the definition of operators that access items of a source analysis situation. The subsequent Section 5.2 introduces navigation step schemas in Subsection 5.2.1 by a generic definition that also comprises navigation guards at schema level. In Subsection 5.2.2, we elaborate on navigation guards at schema level.

5.2 Navigation Step Schemas

In a navigation step, a navigation operator is applied to a source analysis situation and returns a target analysis situation. The invocation of a navigation operator needs actual parameters depending on the operator definition. Such parameters can be fixed to constant values when modeling the navigation step. But it is also useful to introduce unbound variables for navigation operator parameters (unbound parameters). Such parameter values are not fixed at modeling time but at execution time. Analogously to analysis situation schemas, one can think of an instantiation of a navigation step schema. At instantiation time, yet unbound parameters are bound that leads to a navigation step (or navigation step instance) provided that all properties of the source and target analysis situation are also bound to constant values. If all parameters of a navigation step schema and also all properties of the source and target are already bound, we have the special case that the navigation step schema corresponds directly to a navigation step.

In this section, first we give generic definitions of navigation step schemas (including navigation guards) and definitions of instances of navigation step schemas. Specific operators that can be used in navigation guards are introduced in a subsequent subsection including further details about navigation guards. The notion of type-compliant navigation steps is discussed in the last subsection of this section.

5.2.1 Generic Definitions

The following generic definition formalizes the notion of navigation step schemas and can be applied to all types of navigation operators, and regardless whether the source and target are non-comparative or comparative analysis situation schemas.

Definition 5.6. A *navigation step schema* NAV is defined as

- (a) $NAV = (SRC, NavGrd, OP(p_1, \dots, p_q), (\bar{p}_1, \dots, \bar{p}_q), TRG)$ or
- (b) $NAV = (SRC, NavGrd, OP(OP'(p'_1, \dots, p'_q), p_1, \dots, p_q), ((\bar{p}'_1, \dots, \bar{p}'_q), \bar{p}_1, \dots, \bar{p}_q), TRG)$

comprising the following components:

1. SRC is a source analysis situation schema,
2. $NavGrd$ (*navigation guard*) is a boolean expression (also expression `true` allowed which indicates that navigation is always allowed) defined over instances of SRC using operators defined in Table 5.1 of Section 5.2.2,
3. $OP(p_1, \dots, p_q)$ is a navigation operator such that formal parameter p_1 does not represent a non-comparative navigation operator,
4. $\bar{p}_1, \dots, \bar{p}_q$ are constants and variables that represent actual parameters with respect to the list of formal parameters p_1, \dots, p_q of the navigation operator with name OP ,
5. $OP'(p'_1, \dots, p'_{q'})$ is a non-comparative navigation operator,
6. $\bar{p}'_1, \dots, \bar{p}'_{q'}$ are constants and variables that represent actual parameters with respect to the list of formal parameters $p'_1, \dots, p'_{q'}$ of the non-comparative navigation operator with name OP' , and
7. TRG is a target analysis situation schema.

If both SRC and TRG are non-comparative analysis situation schemas, NAV is also called *non-comparative navigation step schema*, else if SRC or TRG (or both) represents a comparative analysis situation schema, NAV is also called *comparative navigation step schema*.

To refer at schema level to operator invocation, one can write $TRG = SRC.[NavGrd] OP(\bar{p}_1, \dots, \bar{p}_q)$ in the case of non-comparative navigation step schemas and $TRG = SRC.[NavGrd] OP(OP'(\bar{p}'_1, \dots, \bar{p}'_{q'}), \bar{p}_1, \dots, \bar{p}_q)$ in the case of comparative navigation step schemas.⁷ In the case that $NavGrd = \text{true}$, one can also omit expression $[NavGrd]$ and write $TRG =$

⁷Note that this notation is not an operator invocation itself. It only refers to an operator invocation at schema level. Only after instantiation where all variables are bound, one obtains an operator invocation. Furthermore, note that instead of formal parameters only actual parameters are given in this notation.

$SRC.OP(\bar{p}_1, \dots, \bar{p}_q)$ and $TRG = SRC.OP(OP'(\bar{p}'_1, \dots, \bar{p}'_{q'}), \bar{p}_1, \dots, \bar{p}_q)$, respectively.⁸

Moreover, we define $Source_{NAV} = SRC$, $NavGrd_{NAV} = NavGrd$, $NavOp_{NAV} = OP(p_1, \dots, p_q)$ or $NavOp_{NAV} = OP(OP'(p'_1, \dots, p'_{q'}), p_1, \dots, p_q)$, respectively, $ActPars_{NAV} = (\bar{p}_1, \dots, \bar{p}_q)$ or $ActPars_{NAV} = ((\bar{p}'_1, \dots, \bar{p}'_{q'}), \bar{p}_1, \dots, \bar{p}_q)$, respectively, and $Target_{NAV} = TRG$.

Furthermore, we allow to use symbol ? to denote a variable (without introducing a variable name). Thus in the case of variables, one can write ? for an actual parameter \bar{p}_i with $1 \leq i \leq q$ and also for an actual parameter \bar{p}'_i with $1 \leq i \leq q'$. \square

This generic definition comprises two cases. Both of them contain a source analysis situation schema, a navigation guard, a navigation operator, actual parameters with respect to the operator's formal parameters, and a target analysis situation schema. The difference in the definition of both cases lies in the first parameter of the navigation operator. In the second case, the first parameter of the navigation operator (which represents a comparative one) defines a non-comparative navigation operator. Analogously, the list of actual parameters (concerning the comparative navigation operator), in turn, contains a list of actual parameters with respect to the included non-comparative navigation step.

In Definition 5.6, navigation guards are introduced at schema level as a part of a navigation step schema. Navigation guards control the application of navigation operators. Finally, after instantiation they are part of navigation steps as introduced in Chapter 4.⁹ Only if the navigation guard of

⁸Casually, because navigation guard **true** means that the instanced navigation step can always be performed, one can think of “a missing navigation guard” in the sense that there is no navigation guard at all—and we also make use of this perception in this thesis. But note, formally, a navigation step schema as well as a navigation step always contain a navigation guard which can be possibly **true**.

⁹In Chapter 4, navigation guards were introduced that only depend on the result set of the source analysis situation. The definition of navigation step schemas introduced in this chapter also allows to use operators that do not examine the result set of the source analysis situation but components of the source analysis situation. In this sense, Table 5.1 of Chapter 5 represents an extension of Table 4.1 in Chapter 4. Finally, note that navigation guards of navigation step schemas are just boolean expressions that become boolean expressions of navigation steps at instantiation time. Thus, the use of navigation

a navigation step evaluates to true, the corresponding navigation operator is invoked (meaning navigation is performed) and the query of the target analysis situation is executed. If the navigation guard evaluates to false, the navigation operator of a navigation step is not invoked meaning that the navigation step is not performed which causes that the query of the target is not executed (no result set of the target analysis situation is created). In Subsection 5.2.2, we present further details about navigation guards with respect to schema and instance level.

Finally, we introduce a generic definition of an instance of a navigation step schema where all free variables are bound. When instantiating a navigation step schema, one obtains a navigation step containing two instances of the source and target analysis situation schemas (the source and the target analysis situation of the navigation step). The navigation guard of the navigation step schema is transferred to the instantiated navigation step.

Definition 5.7. An *instance of navigation step schema* NAV (as presented in Definition 5.6) is a navigation step **nav** defined as

- (a) **nav** = (**src**. $OP(\bar{p}_1, \dots, \bar{p}_q)$, **trg**, $NavGrd$) or
- (b) **nav** = (**src**. $OP(OP'(\bar{p}'_1, \dots, \bar{p}'_{q'}), \bar{p}_1, \dots, \bar{p}_q)$, **trg**, $NavGrd$)

such that

1. **src** is an instance of $Source_{NAV}$,
2. for case (a) with $NavOp_{NAV} = OP(p_1, \dots, p_q)$, $ActPars_{NAV} = (\bar{p}_1, \dots, \bar{p}_q)$, and with $1 \leq i \leq q$, $\bar{p}_i = \bar{p}_i$, if \bar{p}_i is a constant, or, in the case that \bar{p}_i is a variable, \bar{p}_i is a constant to which variable \bar{p}_i is bound,
3. for case (b) with $NavOp_{NAV} = OP(OP'(p'_1, \dots, p'_{q'}), p_1, \dots, p_q)$, $ActPars_{NAV} = ((\bar{p}'_1, \dots, \bar{p}'_{q'}), \bar{p}_1, \dots, \bar{p}_q)$, and with $1 \leq i \leq q'$ and $1 \leq j \leq q$,
 - $\bar{p}'_i = \bar{p}'_i$, if \bar{p}'_i is a constant, or, in the case that \bar{p}'_i is a variable, \bar{p}'_i is a constant to which variable \bar{p}'_i is bound,

guards introduced in Definition 4.3 of Chapter 4 is extended by further operators listed in Table 5.1 of Chapter 5.

- $\bar{\bar{p}}_j = \bar{p}_j$, if \bar{p}_j is a constant, or, in the case that \bar{p}_j is a variable, $\bar{\bar{p}}_j$ is a constant to which variable \bar{p}_j is bound,
4. **trg** is an instance of $Target_{NAV}$, and
 5. $NavGrd = NavGrd_{NAV}$. □

Accordingly to this definition, the instantiation of a navigation step schema can be considered as *schema-compliant* (or *type-compliant*)¹⁰ in the following sense: If a navigation operator of a navigation step schema is applied to an instance of the source analysis situation schema of this navigation step schema provided that the operator's preconditions are satisfied, then the resulting target analysis situation is expected to be an instance of the target analysis situation schema of this navigation step schema such that the operator's postconditions are fulfilled. Moreover, this can be justified formally as follows: In Definition 5.7, analysis situation **src** is an instance of analysis situation schema $Source_{NAV}$ and analysis situation **trg** is an instance of analysis situation schema $Target_{NAV}$. Both analysis situations **src** and **trg** represent the source and target of navigation step **nav** (instance of navigation step schema NAV). Per definition (see Definition 4.3), navigation step **nav** comprises a valid operator invocation such that $\mathbf{trg} = \mathbf{src}.OP(\bar{\bar{p}}_1, \dots, \bar{\bar{p}}_q)$, for case (a), and $\mathbf{trg} = \mathbf{src}.OP(OP'(\bar{\bar{p}}'_1, \dots, \bar{\bar{p}}'_q), \bar{\bar{p}}_1, \dots, \bar{\bar{p}}_q)$, for case (b).

In the following subsections, parts of the previous generic definitions are concretized. Subsection 5.2.2 introduces operators that can be used in boolean expressions for defining navigation guards at schema level. Subsection 5.2.3 deals with type-compliant navigation steps in detail. In this subsection, type checking at schema and instance level is discussed (static and dynamic type checking). Examples of navigation step schemas, graphical representations, and navigation patterns are presented in Section 5.3.

¹⁰We also use the term *type-compliant* as a synonym for *schema-compliant*.

5.2.2 Operators Used in Navigation Guards

The definition of navigation step schemas (Definition 5.6) comprises navigation guards at schema level. A navigation guard represents a boolean expression that is used to control navigation at instance level. In Table 5.1, operators are listed that can be used to form boolean expressions that represent navigation guards. A navigation guard examines properties of a source analysis situation. Thus, in Table 5.1, each operator is prefixed (in an object-oriented style) by a non-comparative source analysis situation (denoted as **as**) or by a comparative source analysis situation (denoted as **cas**), respectively. The left column of Table 5.1 shows the type of analysis situation to which an operator can be applied (non-comparative analysis situation **as** or comparative analysis situation **cas**), the operator name, and the operator's formal parameters (dimension schema D , dimensional predicate P , base measure predicate B , aggregate measure M , aggregate measure predicate F , join condition J , score S , and score predicate SF); the right column comprises the operator definition, i.e., the operator's semantics.

Most operators of Table 5.1 refer to components of an analysis situation. Either a component is returned or a component is evaluated to true or false. For example, operator `as.granLevel(D)` returns granularity level $GranLvl_{as}(D)$ and operator `as.containsSliceCond(D, P)` returns true, if $P \in SliceConds_{as}(D)$. The last four rows of Table 5.1 show operators that return whether a result set after the execution of the query of an analysis situation is empty or not—these are the same operators as listed in Table 4.1 of Chapter 4. Whereas in the last case (examination of the result set), evaluation can only be done after query execution of the corresponding source analysis situation, in the first case, evaluation also can be performed at instantiation time after all free variables of the source analysis situation are bound. Thus, in this case, one can also think of that a navigation guard also controls instantiation. This means that, if the navigation guard is evaluated to false at instantiation time, the instantiation of the navigation step schema can be considered as aborted and no navigation step (no instance) is created. Note that this corresponds to an informal perspective. Formally, a naviga-

tion guard of a navigation step schema always becomes a navigation guard of a corresponding instantiated navigation step. But if this navigation guard is always evaluated to false, this instantiated navigation step will be never performed. Therefore, this leads to an informal perspective, such that one can say that the instantiation itself is not performed or one can say that if the navigation guard of a navigation step schema is false, then no instance can be created.

Moreover, note that navigation guards are used to check conditions about the application domain or they are used to examine the result set after query execution. Navigation guards need not to be used to examine whether preconditions of navigation operators are fulfilled. Preconditions of navigation operators have to be satisfied always independently from navigation guards.¹¹

In the following paragraphs, we shortly describe the operators of Table 5.1. Although, in Table 5.1, we write analysis situations in front of such an operator invocation (separated by a dot symbol¹²), we omit this prefix in the formal notation of an operator invocation (where a navigation guard is enclosed in square brackets) and we also omit this prefix in the graphical representation (presented in Section 5.3) because the source analysis situation (and also the source analysis situation schema) of the operator's application represents an obvious context.

Table 5.1 shows that there are operators that return a component of a dimension qualification (of dimension schema D) of a non-comparative analysis situation **as**: **as.granLevel**(D), **as.diceLevel**(D), and **as.diceNode**(D). Similarly, there are operators that return the same constituents of the context of interest and the context of comparison of a comparative analysis situation **cas**: **cas.granLevelOfCoI**(D), **cas.granLevelOfCoC**(D), **cas.diceLevelOfCoI**(D), **cas.diceLevelOfCoC**(D), **cas.diceNodeOfCoI**(D), and **cas.diceNodeOfCoC**(D).

In the case of slice conditions, one can use different operators that examine whether a dimensional predicate is contained in the set of slice condi-

¹¹Note that although it is not necessary, it is not forbidden that navigation guards examine parts or even the whole precondition of a navigation step.

¹²We use an object-oriented style to indicate the application of such an operator to a non-comparative or comparative analysis situation.

Table 5.1: Operators used in navigation guard expressions and applied to a non-comparative analysis situation **as** or to a comparative analysis situation **cas**

Operator	Definition
<code>as.granLevel(D)</code>	$GranLvl_{\mathbf{as}}(D)$
<code>cas.granLevelOfCoI(D)</code>	$GranLvl_{CoI_{\mathbf{cas}}}(D)$
<code>cas.granLevelOfCoC(D)</code>	$GranLvl_{CoC_{\mathbf{cas}}}(D)$
<code>as.diceLevel(D)</code>	$DiceLvl_{\mathbf{as}}(D)$
<code>cas.diceLevelOfCoI(D)</code>	$DiceLvl_{CoI_{\mathbf{cas}}}(D)$
<code>cas.diceLevelOfCoC(D)</code>	$DiceLvl_{CoC_{\mathbf{cas}}}(D)$
<code>as.diceNode(D)</code>	$DiceNode_{\mathbf{as}}(D)$
<code>cas.diceNodeOfCoI(D)</code>	$DiceNode_{CoI_{\mathbf{cas}}}(D)$
<code>cas.diceNodeOfCoC(D)</code>	$DiceNode_{CoC_{\mathbf{cas}}}(D)$
<code>as.containsSliceCond(D, P)</code>	$P \in SliceConds_{\mathbf{as}}(D)$
<code>cas.containsSliceCondInCoI(D, P)</code>	$P \in SliceConds_{CoI_{\mathbf{cas}}}(D)$
<code>cas.containsSliceCondInCoC(D, P)</code>	$P \in SliceConds_{CoC_{\mathbf{cas}}}(D)$
<code>as.containsBMsCond(B)</code>	$B \in BMsrConds_{\mathbf{as}}$
<code>cas.containsBMsCondInCoI(B)</code>	$B \in BMsrConds_{CoI_{\mathbf{cas}}}$
<code>cas.containsBMsCondInCoC(B)</code>	$B \in BMsrConds_{CoC_{\mathbf{cas}}}$
<code>as.containsAMsr(M)</code>	$M \in AMsrs_{\mathbf{as}}$
<code>cas.containsAMsrInCoI(M)</code>	$M \in AMsrs_{CoI_{\mathbf{cas}}}$
<code>cas.containsAMsrInCoC(M)</code>	$M \in AMsrs_{CoC_{\mathbf{cas}}}$
<code>as.containsFilterCond(F)</code>	$F \in FilterConds_{\mathbf{as}}$
<code>cas.containsFilterCondInCoI(F)</code>	$F \in FilterConds_{CoI_{\mathbf{cas}}}$
<code>cas.containsFilterCondInCoC(F)</code>	$F \in FilterConds_{CoC_{\mathbf{cas}}}$
<code>cas.containsJoinCond(J)</code>	$J \in JoinConds_{\mathbf{cas}}$
<code>cas.containsScore(S)</code>	$S \in Scores_{\mathbf{cas}}$
<code>cas.containsScoreFilter(SF)</code>	$SF \in ScoreFilters_{\mathbf{cas}}$
<code>as.hasResult()</code>	$ResultSet_{\mathbf{as}} \neq \emptyset$
<code>cas.hasResult()</code>	$ResultSet_{\mathbf{cas}} \neq \emptyset$
<code>as.hasNoResult()</code>	$ResultSet_{\mathbf{as}} = \emptyset$
<code>cas.hasNoResult()</code>	$ResultSet_{\mathbf{cas}} = \emptyset$

tions of a non-comparative analysis situation **as** or of the context of interest and the context of comparison of a comparative analysis situation **cas**: **as.containsSliceCond**(D, P), **cas.containsSliceCondInCoI**(D, P), and **cas.containsSliceCondInCoC**(D, P). Again, these operators are applied with respect to a dimension qualification at dimension schema D .

Analogously to slice conditions, in Table 5.1, operators are defined that examine whether an element is in the set of base measure conditions (**as.containsBMSrCond**(B), **cas.containsBMSrCondInCoI**(B), **cas.containsBMSrCondInCoC**(B)), in the set of aggregate measures (**as.containsAMsr**(M), **cas.containsAMsrInCoI**(M), **cas.containsAMsrInCoC**(M)), in the set of filter conditions (**as.containsFilterCond**(F), **cas.containsFilterCondInCoI**(F), **cas.containsFilterCondInCoC**(F)), in the set of join conditions (**cas.containsJoinCond**(J)), in the set of scores (**cas.containsScore**(S)), and in the set of score filters (**cas.containsScoreFilter**(SF)). The last three operators are only applicable to comparative analysis situations.

Finally, in Table 5.1, operators are specified that determine whether a result set of a non-comparative analysis situation **as** or comparative analysis situation **cas** is empty or not: **as.hasResult**(), **cas.hasResult**(), **as.hasNoResult**(), and **cas.hasNoResult**(). These operators do not only require that an analysis situation schema is instantiated but it is also necessary that the query of the corresponding analysis situation is executed and, thus, its result set is available.

The operators listed in Table 5.1 allow to distinguish between two categories of navigation guards. Navigation guards as already presented in Chapter 4 are used to examine the result set of source analysis situations. Such navigation guards must be evaluated after query execution of the source analysis situation. The second category of navigation guards examines the value of components of the source analysis situation of a navigation step. A meaningful application can only be considered in the context of navigation step schemas that use variables in the source analysis situation schema. Such navigation guards can be evaluated at instantiation time and, afterwards, the resulting truth value is given independently of the result set of the query

execution of the source analysis situation. Hence, in the case that such navigation guards evaluate to false, one can omit such instantiated navigation steps at all. Especially in graphical representations, such instantiated navigation steps do not need to be depicted. This means that navigation guards of the second category are only drawn meaningfully at schema level. But note that formally also such navigation guards are always parts of the navigation step.

5.2.3 Type-Compliant Navigation Steps

In Subsection 5.2.1, we introduced the notion of *schema-compliant* (*type-compliant*) navigation steps. The generic definition of an instance of a navigation step schema (Definition 5.7) specifies type-compliant navigation steps with respect to a navigation step schema. Type compliance expresses that if a navigation operator of a navigation step schema is applied to an instance of the source analysis situation schema of this navigation step schema, then the resulting target analysis situation is expected to be an instance of the target analysis situation schema of this navigation step schema.¹³ In this sense, type compliance also can be considered as type safety. Thus, we use the terms *schema compliance*, *type compliance*, and *type safety* synonymously.¹⁴

This section elaborates on checking schema compliance. In this sense, we use especially the notion *type safety*. For checking type safety, each property in the source and target analysis situation has to be checked, whether it violates type safety with respect to the navigation operation of the navigation step. The violation of type safety with respect to properties depends on the pre- and postconditions of a navigation operator together with the frame assumption. Moreover, our claim for checking type safety is that both pre-

¹³In object-oriented programming languages one can formulate a similar requirement: If a method of class *A* with return type of class *B* is called for an instance of class *A*, then the return value of this method call is an instance of class *B*.

¹⁴The consideration of analysis situations and navigation steps at schema level (analysis situation schemas and navigation step schemas) can be compared with classification of objects in object-oriented programming languages realized by classes that also represent (data) types. In this context, we use the term *type* in the notions *type compliance* and *type safety*.

and postconditions of an operator have to be fulfilled.

In this context, one has to distinguish between static and dynamic type safety. Similar to programming languages, *static type safety* (*static type compliance*, *static schema compliance*) can be checked at schema level, whereas *dynamic type safety* (*dynamic type compliance*, *dynamic schema compliance*) just can be checked at instance level.¹⁵ Although navigation guards at schema level (as introduced in Subsection 5.2.2) can be used to examine parts or the whole precondition of a navigation step, the use of navigation guards for the purpose of dynamic type checking is not necessary because if a navigation operator's precondition is not satisfied, no navigation step may be instantiated from the underlying navigation step schema.

A navigation operator changes specific properties of a source analysis situation and transfers these changes to the target analysis situation (see Chapter 4), whereas remaining properties of the source analysis situation are transferred one-to-one (i.e., without changes) to the target accordingly to the frame assumption. At schema level, static type safety can be fully checked, if all properties are constant in both the source analysis situation schema and the target analysis situation schema. In this case, properties that are changed by the navigation operator already have to fulfill the pre- and postcondition of the operator at schema level, and the remaining properties of the source analysis situation schema have to be equal with the corresponding properties of the target analysis situation schema. This checking of pre- and postconditions and the checking of the frame assumption is necessary for providing type checking and can be performed as static type checking for navigation step schemas (i.e., at schema level), if all properties are constant in both the source and the target analysis situation schema.

If variables occur in navigation step schemas, the decision whether static type checking is possible becomes more difficult. Variables in navigation step schemas can appear as properties in the source and target analysis situation schema, but also as actual parameters in the operator invocation. Static type checking can be performed at schema level, if the configuration

¹⁵Similar to programming languages, static type safety in APMN4BI can be checked at compile time, whereas dynamic type safety just can be checked at runtime.

of variables and constants allows to check the navigation operator's pre- and postcondition, and also the frame assumption. If this is not possible, type checking has to be deferred to instantiation time, i.e., only dynamic type checking can be performed. For checking preconditions, only the properties of the source analysis situation schema and the operator's actual parameters have to be taken into account. Properties of the target analysis situation schema are not part of a precondition. On the other side, postconditions can contain properties of the source and target analysis situation schema, and actual parameters. Concerning the frame assumption, only the properties of the source and target analysis situation schema are of interest, but not actual parameters. These considerations are used in the subsequent discussion about type checking including variables.

Table 5.2 summarizes all possible configurations of variables and constants that can be contained as properties of the source and target analysis situation schema, and as actual parameters. The fourth column indicates the kind of type checking (type checking option) that can be performed for the corresponding configuration. The table entry "static" means that static type checking is possible whereas the entry "dynamic" indicates that only dynamic type checking can be performed, but not static type checking. The entry "not reasonable" indicates a configuration that is not reasonable.

The first row of Table 5.2 represents the configuration as previously discussed in this subsection where all properties of the source and target analysis situation schema, and all actual parameters are constants, i.e., no variables occur. As already explained, this configuration allows static type checking.

The second row of Table 5.2 represents the configuration that all properties of the source analysis situation schema and all actual parameters of the navigation operator are constant but one or more variables occur as properties in the target analysis situation schema. In this case, the operator's precondition can be checked at schema level. Although the operator's postcondition cannot be checked at schema level with respect to the decision whether the postcondition is satisfied or not (because the target analysis situation schema comprises variables), examination is possible whether constants violate the postcondition and, on the other side, the values for variables are

Table 5.2: Type checking options depending on the configuration of constants and variables as properties of the source analysis situation schema, as actual parameters, and as properties of the target analysis situation schema.

Nº	Source Items	Actual Parameters	Target Items	Type Checking Option
1	all constant	all constant	all constant	static
2	all constant	all constant	some variable	static ^a
3	all constant	some variable	all constant	not reasonable ^b
4	all constant	some variable	some variable	dynamic
5	some variable	all constant	all constant	dynamic
6	some variable	all constant	some variable	dynamic
7	some variable	some variable	all constant	not reasonable ^c
8	some variable	some variable	some variable	dynamic

^aThe assignment of variables in the target analysis situation schema is determined by the navigation operator of the navigation step schema accordingly to the operator definitions in Chapter 4. Together with the fact that all properties of the source analysis situation schema and all actual parameters are constant, all checks concerning type safety can be performed at schema level. Thus, static type checking is possible for this configuration.

^bIf all properties of the source and target analysis situation schema are constant, it is not reasonable to use variables as actual parameters because only one specific (and, thus, constant) configuration of actual parameters has to be used to transform and to transfer the property values of the source to the property values of the target by the navigation operator. Thus, this configuration should be modeled like the configuration in the first row where also all actual parameters are constant.

^cThis is a similar configuration like in the third row, i.e., if all properties of the target analysis situation schema are constant, it is not reasonable to use variables as actual parameters because only one specific (and, thus, constant) configuration of actual parameters has to be used to transform and to transfer the property values of the source to the property values of the target by the navigation operator. Thus, this configuration should be modeled like the configuration in the fifth row where also all actual parameters are constant.

determined by the postcondition (together with the frame assumption) at instance level. Hence, if one can check at schema level that the postcondition together with the frame assumption is not violated, one also can rely (at schema level) on that the postcondition is satisfied at instance level. In this sense, static type checking can be performed.

The configuration in the third row of Table 5.2 is not a reasonable one. If all properties in the source and target analysis situation are constant, there is no reason that one should use variables as actual parameters for the operator invocation. Because source and target are fully determined at schema level, also the operator invocation is fully determined at schema level which means that also all actual parameters are determined by constant values and, thus, there is no need for variables as actual parameters. Such a configuration has to be avoided in the modeling process, i.e., such a configuration can be avoided at schema level which also corresponds in some sense to a kind of static type checking.

In row number four of Table 5.2, all properties of the source analysis situation schema are constant but variables occur as actual parameters of the operator as well as properties of the target analysis situation schema. Because there are variables as actual parameters, the precondition of the navigation step schema cannot be checked at schema level but only at instance level (after variable binding). Thus, static type checking is not possible, only dynamic type checking can be performed.

Rows 5–8 in Table 5.2 comprise the configurations where variables are used as properties of the source analysis situation schema. Analogously to row number four, the precondition of the navigation step schema cannot be checked at schema level.¹⁶ Only after binding the variables of the source analysis situation schema and variables of the navigation operator's actual parameter list, the precondition can be checked at instance level (dynamic type checking). Static type checking is not possible in these situations.

Moreover, row seven of Table 5.2 represents a configuration that is not

¹⁶Note that checking of the precondition including the frame assumption also requires that after instantiation of a navigation step schema the resulting source analysis situation corresponds to Definition 3.2 and 3.4, respectively.

reasonable (similar to the third row of Table 5.2). If all properties in the target analysis situation schema are constant, there is no reason to use variables as actual parameters for the operator invocation. The actual parameters are determined by the constant properties of the target analysis situation schema. Thus, such a situation should be modeled accordingly to the fifth row of Table 5.2, i.e., such a configuration can be avoided at schema level which also corresponds in some sense to a kind of static type checking.

Note that in the configuration of row five in Table 5.2, all actual parameters and all properties of the target analysis situation schema are constant. In this case, one could challenge why there are variables in the source analysis situation schema, i.e., if all actual parameters and all properties of the target analysis situation schema are constant, then one could argue that also all properties of the source analysis situation schema should be constant. But this argument does not hold anymore, if one considers sequences of navigation step schemas (see Chapter 6), i.e., if a preceding navigation step schema requires variables in the target analysis situation schema, then in the subsequent navigation step schema this target represents a source analysis situation schema with variables as indicated in row five of Table 5.2.

5.3 Example of Navigation Step Schemas

In this section, several examples of navigation step schemas are presented. Navigation step schemas always have navigation guards per definition. The following subsection demonstrates navigation step schemas that only have navigation guards containing boolean expression `true`. Casually, for such navigation step schemas, we use the phrase “navigation step schemas without navigation guards”. In this case, navigation guards are not depicted graphically. Afterwards, navigation step schemas with navigation guards (meaning navigation guards containing boolean expressions other than expression `true`) are presented in another subsection. The last subsection comprises exemplary navigation patterns that can be used as general templates.

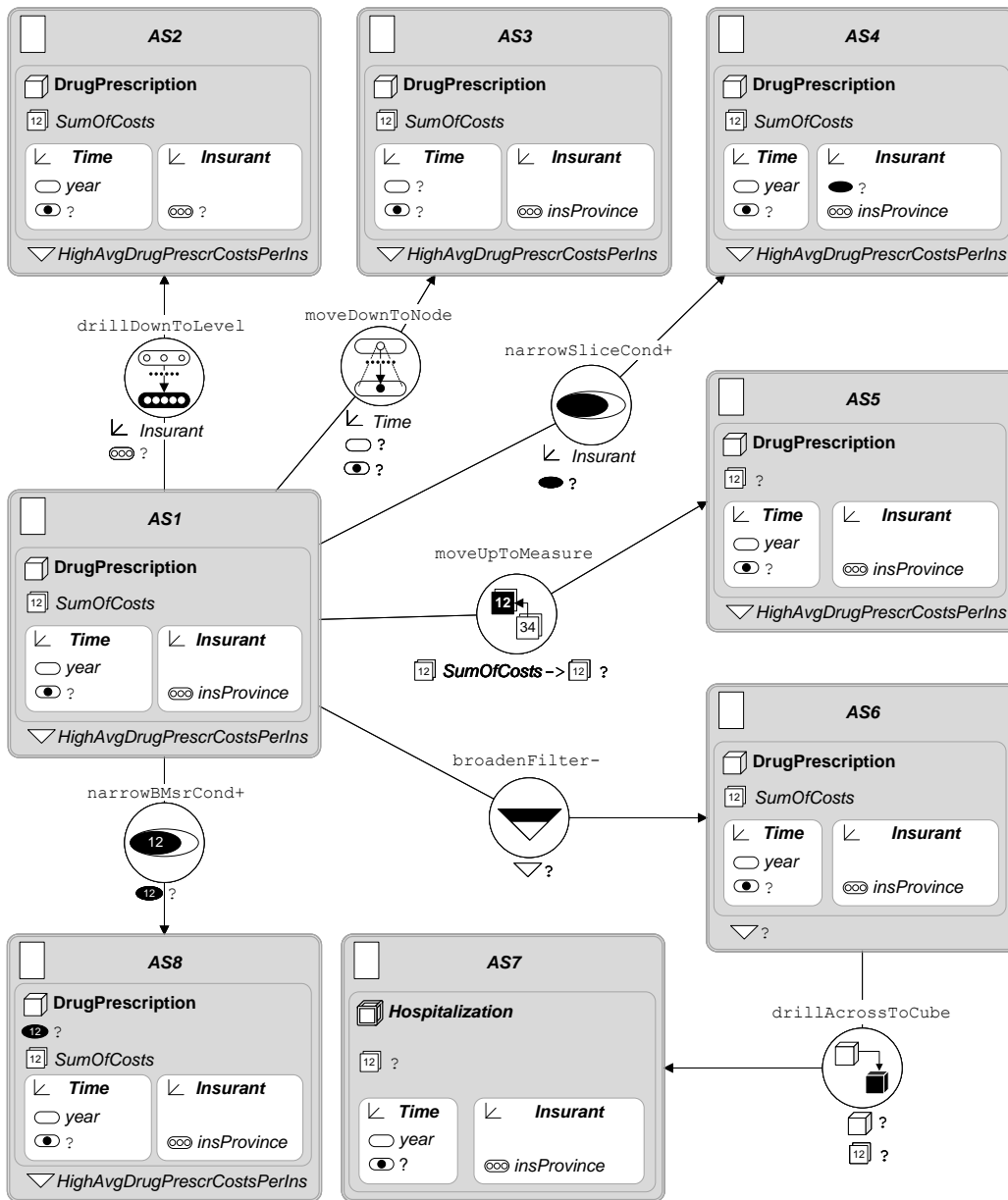


Figure 5.9: Navigation step schemas with unbound variables

5.3.1 Examples without Navigation Guards

In Figure 5.9, several non-comparative navigation step schemas without navigation guards are depicted.¹⁷ The navigation step schema from *AS1* to *AS2* comprises an unbound variable for the dice node in source analysis situation schema *AS1*, an unbound parameter for granularity level in dimension schema *Insurant* of operator invocation `drillDownToLevel`, and two unbound variables for dice node in *Time* and granularity level in *Insurant* of the target analysis situation. Formally, this navigation step schema can be written following: $(AS1, \text{true}, \text{drillDownToLevel}(D, G), (Insurant, ?), AS2)$. It induces the following operator invocation at schema level: $AS2 = AS1.\text{drillDownToLevel}(Insurant, ?)$.

Concerning type compliance, one can see that in navigation step schema from *AS1* to *AS2* only dynamic type checking but not static type checking is possible because there is a variable in source *AS1* and another one as an actual parameter, i.e., the operator's precondition and postcondition (including the frame assumption) cannot be checked at schema level. Thus, accordingly to Table 5.2, only dynamic type checking can be performed. This argumentation concerning type compliance can also be used with respect to the other navigation step schemas depicted in Figure 4.1. All navigation step schemas of Figure 4.1 comprise variables as properties of the source analysis situation schema and as actual parameters.¹⁸ Hence, in Figure 4.1, there is no navigation step schema where static type checking can be performed but only dynamic type checking is possible.

In Figure 4.1 of Chapter 4, navigation step from non-comparative analysis situation *as1* to non-comparative analysis situation *as2* can be considered as an example of an instance of navigation step schema from *AS1* to *AS2*. The

¹⁷Casually, navigation step schemas having boolean expression `true` as navigation guard are considered as navigation step schemas having no navigation guard. Formally, each navigation step schema comprises a navigation guard that can be defined as boolean expression `true`.

¹⁸Note that the navigation step schemas of Figure 4.1 also contain variables in the target analysis situation schemas but variables in the source analysis situation schemas and variables as actual parameters are already crucial that static type checking cannot be performed.

variable for the dice node with respect to dimension schema *Time* of *AS1* is bound to 2016 (leading to *as1* which is an instance of *AS1*), the variable for the second actual parameter of navigation operator *drillDownToLevel* is bound to dimension level *insDistrict*, and the variable for dice node again with respect of dimension schema *Time* and the variable for granularity level concerning dimension schema *Insurant* both of target analysis situation schema *AS2* are bound to dimension node 2016 and to dimension level *insDistrict*, respectively (leading to *as2* which is an instance of *AS2*). This instantiation leads to a valid operator invocation ($as2 = as1.drillDownToLevel(Insurant, insDistrict)$) and also to a type-compliant (schema-compliant) navigation step with respect to navigation step schema from *AS1* to *AS2*.

The second example in Figure 5.9 shows a navigation step schema using navigation operator *moveDownToNode*: (*AS1*, *true*, *moveDownToNode(D, L, N)*, (*Time*, *?*, *?*), *AS3*). At schema level, the operator invocation can be written as $AS3 = AS1.moveDownToNode(Time, ?, ?)$. Figure 4.1 of Chapter 4 presents an example of an instance of this navigation step schema where the variable for the second actual parameter of operator *moveDownToNode* and the variable for the dice level of dimension qualification of dimension schema *Time* of analysis situation *as3* is bound to *quarter*, and the third actual parameter of operator *moveDownToNode* and the corresponding dice node of analysis situation *as3* is bound to 2016Q1. This results to the following valid operator invocation: $as3 = as1.moveDownToNode(Time, quarter, 2016Q1)$.

Another example of a navigation step schema depicted in Figure 5.9 comprises source analysis situation schema *AS1*, target analysis situation schema *AS4*, and navigation operator *narrowSliceCond+*. The second actual parameter of operator *narrowSliceCond+* represents a variable for a set of dimensional predicates with respect to dimension schema *Insurant* and the slice condition of the dimension qualification of dimension schema *Insurant* of the target analysis situations schema *AS4* is also defined as a variable. Formally, this navigation step schema and the induced operator invocation at schema level can be written in the following way: (*AS1*, *true*, *narrowSliceCond+(D, P)*, (*Insurant*, *?*), *AS4*) and $AS4 = AS1.narrowSliceCond+(Insurant, ?)$. Again, Figure 4.1 contains an example of an instance of this navigation step

schema with `as1` as source analysis situation and `as4` as target analysis situation. In this example both variables are bound to a set containing the single dimensional predicate *InsInRuralDistrict* leading to valid operator invocation: `as4 = as1.narrowSliceCond+(Insurant, {InsInRuralDistrict})`.

Navigation operator `moveUpToMeasure` is used in Figure 5.9 for navigation step schema containing source analysis situation schema *AS1* and target analysis situation schema *AS5*. This navigation step schema and its induced operator invocation at schema level can be written formally as $(AS1, \text{true}, \text{moveUpToMeasure}(M_{old}, M_{new}), (SumOfCosts, ?), AS5)$ and $AS5 = AS1.\text{moveUpToMeasure}(SumOfCosts, ?)$. The second actual parameter of operator `moveUpToMeasure` and the set of aggregate measures of target analysis situation schema *AS5* are defined as variables. Navigation step from source analysis situation `as1` to target analysis situation `as5` in Figure 4.1 represents an example of an instance of this navigation step schema where the variable of the second actual parameter of operator `moveUpToMeasure` is bound to aggregate measure *AvgCostsPerInsurant* and the variable for the set of aggregate measures of target analysis situation *AS5* is bound to set $\{AvgCostsPerInsurant\}$. This instantiation yields valid operator invocation `as5 = as1.moveUpToMeasure(SumOfCosts, AvgCostsPerInsurant)`.

Figure 5.9 comprises two navigation step schemas that use navigation operator `broadenFilter-` and navigation operator `narrowBMsCond+`, respectively. Again, analysis situation schema *AS1* represents the source of both navigation step schemas. Beside the variable contained in the source and target analysis situation schema concerning the dice node of dimension qualification with respect to dimension schema *Time*, there are further variables concerning the operators' single parameter (for the set of aggregate measure predicates and for the set of base measure predicates), and the set of filter conditions and the set of base measure conditions of the target. Formally, these navigation step schemas can be written as $(AS1, \text{true}, \text{broadenFilter-}(F), (?), AS6)$ and $(AS1, \text{true}, \text{narrowBMsCond+}(B), (?), AS8)$, and the induced operator invocation at schema level can be written as $AS6 = AS1.\text{broadenFilter-}(?)$ and $AS8 = AS1.\text{narrowBMsCond+}(?)$. Figure 4.1 comprises examples of instances of both navigation step schemas

where the corresponding variables are bound to set $\{HighDrugPrescrCostsPerIns\}$ and to set $\{HighCostsPerUnit\}$, respectively. These instantiations result in the following valid operator invocations: $as6 = as1.broadenFilter-(\{HighDrugPrescrCostsPerIns\})$ and $as8 = as1.narrowBMsrCond+(\{HighCostsPerUnit\})$.

The last example of Figure 5.9 represents a navigation step schema that changes to another cube schema by operator `drillAcrossToCube`. Source analysis situation schema *AS6* is grounded on cube instance `DrugPrescription` which is an instance of cube schema *DrugPrescription*.¹⁹ The actual parameter list of operator `drillAcrossToCube` comprises two variables, one variable for the cube instance and the other one for the set of aggregate measures. Target analysis situation schema *AS7* contains three variables: One variable refers to the dice node of the dimension qualification concerning dimension schema *Time* and the second variable refers to the set of aggregate measure. Both variables are denoted in Figure 5.9 by symbol $?$. The third variable is depicted implicitly and represents a variable for a cube instance that must belong to cube schema *Hospitalization*. In the graphical representation, we use the double-edged cube-symbol to denote the cube schema (with cube schema name *Hospitalization* written in *slant* font). This specifies implicitly that the cube instance of analysis situation schema *AS7* represents a variable.²⁰

Formally, this navigation step schema can be written in the following way: $(AS6, true, drillAcrossToCube(c, B, M, F), (?, \emptyset, ?, \emptyset), AS7)$. Note that the actual parameter for the set of base measures and the set of filter conditions are set to empty set (considered as constants). Analysis situation schema *AS7* can be written as $(Hospitalization, ?, \emptyset, ?, DQS, \emptyset)$ where $DQS = \{DQ_{Time}, DQ_{Insurant}, DQ_{Hospital}\}$ represents the set of dimen-

¹⁹Note that in this example we use the same name for cube instance and cube schema. Both can only be distinguished by different font styles: cube instance `DrugPrescription` is written in sans serif font and cube schema *DrugPrescription* in *slant* font.

²⁰Note, as a difference, the graphical representation of analysis situation schema *AS6* contains a single-edged cube symbol that denotes constant cube instance `DrugPrescription` (written in sans serif font) which belongs to cube schema *DrugPrescription*. Similarly, the first depicted actual parameter of operator `drillAcrossToCube` in Figure 5.9 is graphically prefixed by a single-edged cube symbol because the variable also refers to a cube instance.

sion qualifications defined as $DQ_{Time} = (Time, year, ?, \emptyset, top)$, $DQ_{Insurant} = (Insurant, top, all, \emptyset, insProvince)$, and $DQ_{Hospital} = (Hospital, top, all, \emptyset, top)$. The induced operator invocation at schema level can be written as $AS7 = AS6.drillAcrossToCube(?, \emptyset, ?, \emptyset)$.

In Figure 4.1, the navigation step from analysis situation `as6` to analysis situation `as7` represents an example of an instance of the navigation step schema from analysis situation schema $AS6$ to analysis situation schema $AS7$. One variable of the operator call is bound to cube instance `Hospitalization` (an instance of cube schema $Hospitalization$) and the second variable is bound to the set of aggregate measures $\{AvgCostsPerInsurant, AvgCostsPerDay\}$. This instantiation yields the following valid operator invocation: `as7 = as6.drillAcrossToCube(Hospitalization, \emptyset , $\{AvgCostsPerInsurant, AvgCostsPerDay\}$, \emptyset)`. The variables of the target are bound to dice node `2016` (with respect to dimension schema $Time$), to cube instance `Hospitalization`, and to the set of aggregate measures $\{AvgCostsPerInsurant, AvgCostsPerDay\}$.

Figures 5.10 and 5.11 comprise examples of navigation step schemas that can be used to compare the various configurations (rows) in Table 5.2 concerning type checking options. Examples where all properties of the source analysis situation schema are constant, are included in Figure 5.10 (corresponding to rows 1–4 in Table 5.2) and examples where the source analysis situation schema contains a variable property are included in Figure 5.11 (corresponding to rows 5–8 in Table 5.2).

In Figure 5.10, all examples of navigation step schemas use the same source analysis situation schema $AS1$ with constant items and the navigation operator `drillDownToLevel` with respect to dimension schema $Insurant$. Navigation step schema from source analysis situation schema $AS1$ to target analysis situation schema $AS2$ comprises no variables but only constants as items in the source and the target, and also as actual parameters. This corresponds to the first row of Table 5.2. Static type checking is possible in this case. The navigation step schema represents the only instantiable navigation step²¹ that performs a drill-down from dimension level $insProvince$ to

²¹Such a navigation step schema can be compared to a singleton class in object-oriented

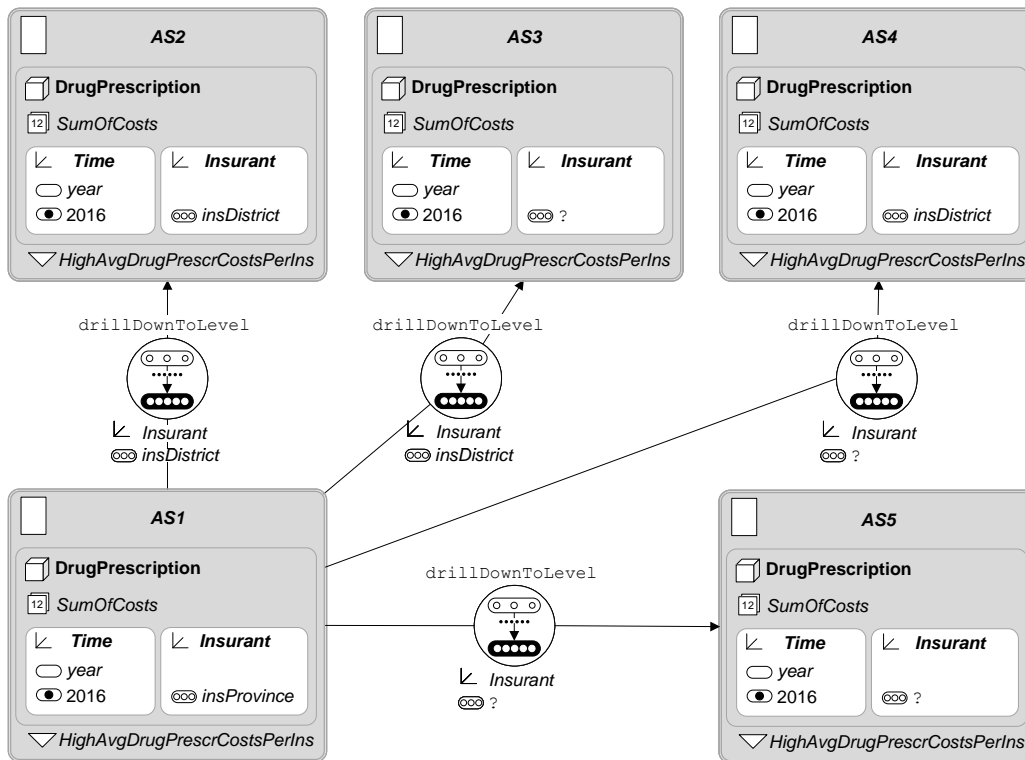


Figure 5.10: Navigation step schemas with constant properties in the source analysis situation schema and type checking options

dimension level *insDistrict*.

The second example of Figure 5.10 comprises the navigation step schema from source analysis situation schema *AS1* to target analysis situation schema *AS3* where only the granularity level of dimension schema *Insurant* of *AS3* is a variable which corresponds to the second row of Table 5.2. This variable is uniquely determined by the actual and constant parameter *insDistrict* which can be examined at schema level, i.e., static type checking can be performed. In this case, one could ask whether it is meaningful to use a variable for this property because it can obtain only one specific value (dimension level *insDistrict*) by the operator *drillDownToLevel*. In Chapter 6, we will see that a target analysis situation schema can be used as a target analysis

languages.

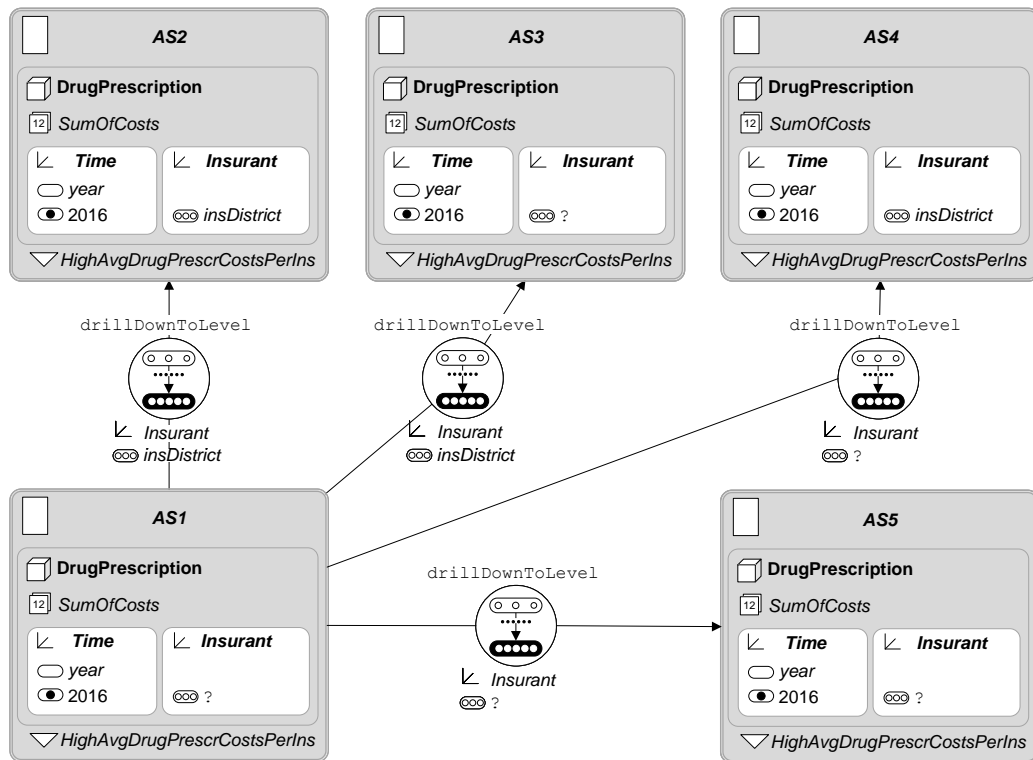


Figure 5.11: Navigation step schemas with a variable property in the source analysis situation schema and type checking options

situation schema of another navigation step schema and this navigation step schema could, for instance, also require variables in the target. Thus, this example of a navigation step schema represents a reasonable one.

Although the previous example of Figure 5.10 can be justified as a meaningful one, navigation step schema from *AS1* to *AS4* is not reasonable (see the third row of Table 5.2). Because of the constants of the source and target analysis situation schemas (*insProvince* and *insDistrict*), there is only one meaningful value for the variable actual parameter, namely, *insDistrict* and there are no other reasons to justify this variable. Thus, this navigation step schema is not reasonable. For properly modeling, one should replace this variable by constant *insDistrict* as actual parameter which, afterwards, represents the same as navigation step schema from *AS1* to *AS2*.

The last example of Figure 5.10 presents the navigation step schema from

source analysis situation schema *AS1* to target analysis situation schema *AS5* that comprises a variable as actual parameter and a variable for the granularity level of dimension schema *Insurant* in the target analysis situation schema. This corresponds to the configuration of the fourth row of Table 5.2 that only allows dynamic type checking at instantiation time. For example, binding both variables to dimension level *insDistrict* returns a valid navigation step that represents an instance of this navigation step schema. The binding of both variables to dimension level *insurant* also leads to a type-compliant navigation step. On the other side, if both variables are bound to dimension level *top*, the precondition of operator `drillDownToLevel` is violated resulting in an invalid navigation step.

Analogously to the previous examples of Figure 5.10, we discuss the examples of Figure 5.11 where there is a variable for the granularity level of dimension schema *Insurant* in source analysis situation schema *AS1*. In the first example of Figure 5.11, navigation step schema from source *AS1* to target *AS2* only comprises a variable in the source analysis situation schema *AS1* corresponding to the fifth row of Table 5.2. To obtain a valid navigation step, this variable can be bound to dimension level *insProvince* or to dimension level *top*. This type checking cannot be performed at schema level but at instantiation time meaning that only dynamic type checking is possible.

Navigation step schema from *AS1* to *AS3* represents the second example of Figure 5.11 and is related to the sixth row of Table 5.2. Target analysis situation schema *AS3* contains a variable for the granularity level in dimension schema *Insurant*. Dimension level *insDistrict* represents a constant actual parameter of operator `drillDownToLevel` which, finally, also binds the variable of the target analysis situation schema to dimension level *insDistrict*. Nevertheless, type checking can only be performed at instantiation time (only dynamic type checking is possible) because also the variable in the source analysis situation has to be bound. For instance, if this variable in the source is bound to *insProvince* or to *top*, the resulting navigation steps are valid. On the other side, if this variable is bound to *insurant*, the operator's precondition is violated resulting in an invalid navigation step.

The third example of Figure 5.11 represents the navigation step schema

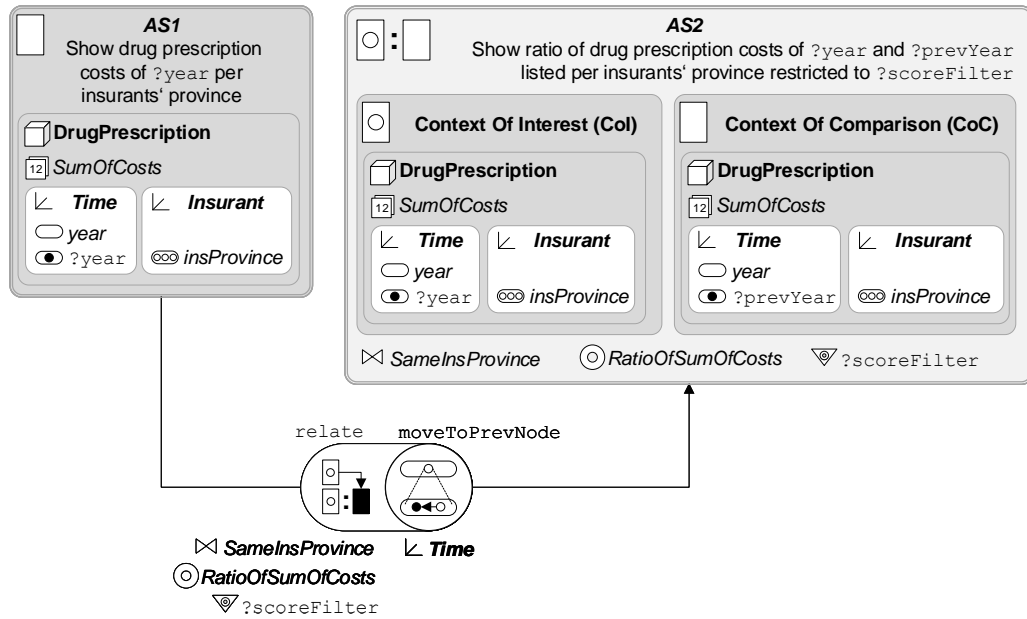


Figure 5.12: Navigation step schema with unbound variables involving a comparative analysis situation

from source *AS1* to target *AS4*. It contains a variable as an actual parameter for the granularity level of dimension schema *Insurant* but the corresponding target property is already set to constant value *insDistrict*. As discussed accordingly to the seventh row of Table 5.2 such configurations are not reasonable.

Similar to the examples of Figure 5.9, the last navigation step schema from *AS1* to *AS5* comprises variables in the source and target analysis situation schema as well as in the navigation operator's actual parameter list. This configuration is presented in the last row of Table 5.2. Of course, also in this case, only dynamic type checking can be performed at instantiation time.

As another example, Figure 5.12 demonstrates a comparative navigation step schema that involves a non-comparative source analysis situation schema and a comparative target analysis situation schema. Additionally, this example presents named variables to increase readability. Source *AS1* contains a variable for the dice node with respect to dimension qualification of dimension

schema *Time*. This variable is named as `?year`. The actual parameter list of the operator call contains named variable `?scoreFilter` and target analysis situation schema *AS2* comprises named variables `?year`, `?prevYear`, and `?scoreFilter`. In the graphical representation, we also allow to use named variables in the description of an analysis situation schema. At instantiation time this variables can be used in the generation of a graphical representation of an analysis situation instance to insert the specific values into the description. Note again, named variables can be used for better readability but semantics must be established by the navigation step itself independently from a specific variable name. In the example of Figure 5.12, the variables for the dice node in the source and for the dice node in the context of interest of the target obtain name `?year` which is meaningful because both dice nodes refer to dice level *year* of the corresponding dimension qualifications. Variable name `?prevYear` (as an abbreviation for previous year) in the context of comparison of the target represents a meaningful name because the application of navigation operator `moveToPrevNode` ensures that after instantiation the value of variable `?prevYear` corresponds to the previous year of the value of variable `?year`. In the case of variable `?scoreFilter` used in the operator call and variable `?scoreFilter` used in the target, the same name is applied to different variables which also makes sense because operator `relate` transfers the set of score predicates of the actual parameter list to the set of score filters of the target. Thus, the operator ensures that both variables are bound to the same value at instantiation time.

The second special feature of the example of Figure 5.12 concerns the fact that the depicted navigation step schema represents a comparative navigation step schema. Formally, this navigation step schema can be written accordingly to Definition 5.6 as $(AS1, \text{true}, \text{relate}(\text{moveToPrevNode}(D), J, S, SF), ((Time), \{SameInsProvince\}, \{RatioOfSumOfCosts\}, ?scoreFilter), AS2)$. Non-comparative source analysis situation schema *AS1* is defined as $AS1 = (DrugPrescription, DrugPrescription, \emptyset, \{SumOfCosts\}, DQS, \emptyset)$ with $DQS = \{DQ_{Time}, DQ_{Insurant}, DQ_{Doctor}, DQ_{Drug}\}$ with $DQ_{Time} = (Time, year, ?year, \emptyset, top)$, $DQ_{Insurant} = (Insurant, top, all, \emptyset, insProvince)$, $DQ_{Doctor} = (Doctor, top, all, \emptyset, top)$, and $DQ_{Drug} = (Drug, top, all, \emptyset, top)$. Com-

parative target analysis situation schema $AS2$ is formally specified as $AS2 = (AS^I, AS^C, \{SameInsProvince\}, \{RatioOfSumOfCosts\}, ?scoreFilter)$ with context of interest $AS^I = AS1$ and, at schema level, the context of comparison AS^C is also equal to $AS1$ except that syntactically (only for better readability) variable name $?prevYear$ is used instead of variable name $?year$.^{22,23}

Concerning type compliance in the example of Figure 5.12, one can see that there are variables in the source analysis situations schema, in the list of actual parameters, and in the target analysis situation schema. Thus, accordingly to Table 5.2, dynamic type checking can be performed but not static type checking.

Figure 4.2 shows an instance of navigation step schema of Figure 5.12. Variable $?year$ of source analysis situation schema $AS1$ is bound to value 2016, variable $?scoreFilter$ of the invocation of operator `relate` is bound to \emptyset , and variables $?year$, $?prevYear$, and $?scoreFilter$ of target analysis situation schema $AS2$ are bound to 2016, 2015, and \emptyset , respectively. This yields valid operator invocation $as2 = as1.relate(moveToPrevNode(Time), \{SameInsProvince\}, \{RatioOfSumOfCosts\}, \emptyset)$.

In Figure 5.13, one non-comparative analysis situation schema named as *DrugCosts* and two different navigation step schemas are depicted. It is a special example in the sense that the single analysis situation schema is used as source and target of both navigation step schemas. Formally, the first navigation step schema can be written as $(DrugCosts, true, moveToNode(D, L, N), (Time, ?, ?), DrugCosts)$ and the second one as $(DrugCosts, true, drillDownOneLevel(D), (Insurant), DrugCosts)$. The non-comparative analysis situation schema contains three variables and can be formalized as $DrugCosts = (DrugPrescription, DrugPrescription, \emptyset, \{SumOfCosts\}, \{DQ_{Time}, DQ_{Insurant}, DQ_{Doctor}, DQ_{Drug}\}, \emptyset)$ with $DQ_{Time} = (Time, ?, ?, \emptyset, top)$, $DQ_{Insurant} = (Insurant, top, all, \emptyset, ?)$, $DQ_{Doctor} = (Doctor, top, all, \emptyset, top)$, and $DQ_{Drug} = (Drug, top, all, \emptyset, top)$. At schema level, the operator invo-

²²If, in this example, only unnamed variables (only symbol $?$) were used, both AS^I and AS^C would have been also syntactically equal.

²³Note that although AS^I and AS^C are equal at schema level, both contexts are always different at instance level (year versus previous year).

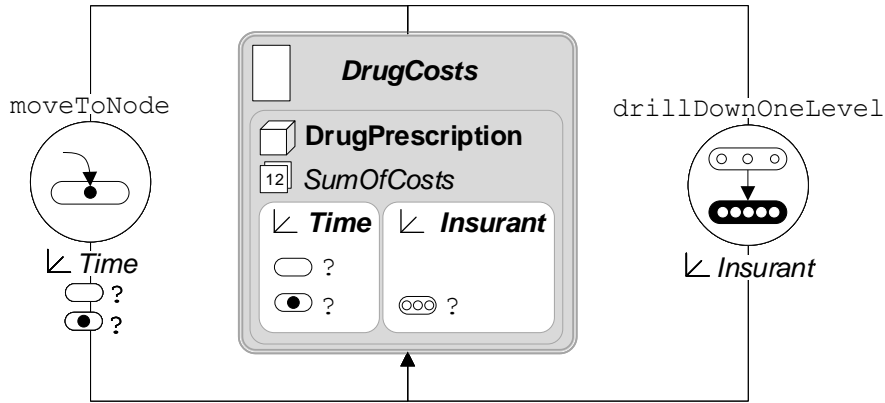


Figure 5.13: Navigation step schemas with equal source and target

cations are written following: $DrugCosts = DrugCosts.moveToNode(Time, ?, ?)$ and $DrugCosts = DrugCosts.drillDownOneLevel(Insurant)$.

Note that although both navigation step schemas in Figure 5.13 have analysis situation schemas $DrugCosts$ as source and target, at instance level, one gets different analysis situation instances as source and target. For example, if $as1$ is an instance of $DrugCosts$ where the variables are bound such that $DiceLvl_{as1}(Time) = year$, $DiceNode_{as1}(Time) = 2017$, and $GranLvl_{as1}(Insurant) = insProvince$, if the variables of the invocation of operator $moveToNode$ are bound to dimension level $year$ and dimension node 2015 , and if a valid invocation yields $as2 = as1.moveToNode(Time, year, 2015)$, we obtain a navigation step $(as1.moveToNode(Time, year, 2015), as2)$ which represents an instance of the navigation step schema with respect to operator $moveToNode$. Moreover, if $as3 = as2.drillDownOneLevel(Time)$ is a valid operator invocation, we obtain a navigation step $(as2.drillDownOneLevel(Time), as3)$ that can be considered as an instance of the navigation step schema with respect to operator $drillDownOneLevel$. All three analysis situations are different instances of analysis situation schema $DrugCosts$: $DiceNode_{as1}(Time) = 2017$, $DiceNode_{as2}(Time) = 2015$, $GranLvl_{as1}(Insurant) = GranLvl_{as2}(Insurant) = insProvince$, and $GranLvl_{as3}(Insurant) = insDistrict$.

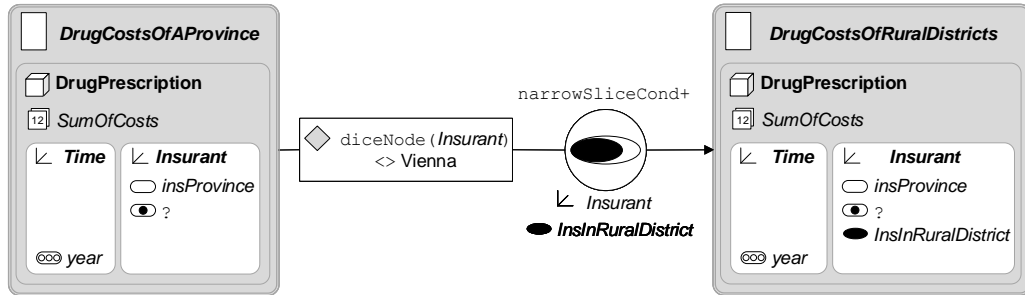


Figure 5.14: Example with navigation guard using operator `diceNode`

5.3.2 Examples with Navigation Guards

In this subsection, examples of navigation step schemas containing navigation guards are presented. Note that, formally, every navigation step schema has a navigation guard per definition. A navigation step schema without a navigation guard can be considered as a navigation step schema having navigation guard specified by boolean expression `true` (see Section 5.2).

Figure 5.14 shows a navigation step schema comprising non-comparative source analysis situation schema *DrugCostsOfAProvince* and non-comparative target analysis situation schema *DrugCostsOfRuralDistricts*. Source and target are linked by operator `narrowSliceCond+` that is preceded by a navigation guard. Graphically, navigation guards are depicted as rectangles containing a diamond in the left upper corner as a pictogram. In the case that a navigation guard represents boolean expression `true`, no rectangle is drawn at all and, in common usage, we also say that there is no navigation guard at all, although, with respect to the formal definition, boolean expression `true` also represents actually a specific navigation guard.

Source analysis situation schema *DrugCostsOfAProvince* can be used for instantiating non-comparative analysis situations to list the sum of drug costs of a province (given by unbound dice node variable with respect to dimension schema *Insurant*) per year. By applying navigation operator `narrowSliceCond+` the target is restricted to rural districts by using dimensional predicate *InsInRuralDistrict*. An instance of this navigation step schema only makes sense, if the dice node variable is bound to a province

having rural districts. It is common knowledge that Vienna as Austria's capital city which is also a province has no rural districts. Thus instances of this navigation step schema are controlled by a navigation guard with boolean expression `diceNode(Insurant) <> Vienna`. The navigation to the target is only performed, if other Austrian provinces except Vienna are selected as dice node. This can be considered as an example how navigation guards can be applied to model additional business knowledge to control navigation.

Formally, the navigation step schema in Figure 5.14 can be written as $(DrugCostsOfAProvince, \text{diceNode}(Insurant) \langle \rangle Vienna, \text{narrowSliceCond}+(D, P), (Insurant, \{InsInRuralDistrict\}), DrugCostsOfRuralDistricts)$ and the operator invocation at schema level as $DrugCostsOfRuralDistricts = DrugCostsOfAProvince.[\text{diceNode}(Insurant) \langle \rangle Vienna] \text{narrowSliceCond}+(Insurant, \{InsInRuralDistrict\})$. The navigation guard is put in square brackets and placed previous to operator name `narrowSliceCond+`.

If `DrugCostsOfUpperAustria` represents an instance of non-comparative analysis situation schema *DrugCostsOfAProvince* and if `DrugCostsOfRuralDistrictsInUpperAustria` represents an instance of non-comparative analysis situation schema *DrugCostsOfRuralDistricts* such that the single variables of both instances are bound to `UpperAustria`, then operator invocation `DrugCostsOfUpperAustria.narrowSliceCond+(Insurant, {InsInRuralDistrict})` yields navigation step $(DrugCostsOfUpperAustria.narrowSliceCond+(Insurant, \{InsInRuralDistrict\}), DrugCostsOfRuralDistrictsInUpperAustria)$ which is an instance of the navigation step schema depicted in Figure 5.14 because the navigation guard evaluates to true. Otherwise, if the variables of the source and target analysis situation schemas are bound to `Vienna` no navigation step is created because the navigation guard evaluates to false due to abortion of the instantiation process as mentioned in Subsection 5.2.2.²⁴

Figure 5.15 shows another example of a navigation step schema compris-

²⁴As mentioned in Subsection 5.2.2, this corresponds to a casual perspective. Formally, if the variables are bound to `Vienna`, the resulting instance (navigation step) comprises navigation guard `diceNode(Insurant) <> Vienna` that yields condition `Vienna ≠ Vienna` which always evaluates to false. Thus, this navigation step is never performed and, therefore, one can think of a navigation step that does not exist at all, i.e., no instance of the navigation step schema in Figure 5.14 is created.

ing a navigation guard that contains operator `hasResult`. Both source analysis situation schema *AS1* and target analysis situation schema *AS2* have two variables in the context of interest and in the context of comparison. The variables refer to dice nodes with respect to dimension qualifications of dimension schema *Time*. If the variables of the context of interest are bound to year 2016 and if the variables of the context of comparison are bound to year 2015, one obtains an instance that corresponds to the navigation step depicted in Figure 4.12 of Chapter 4. As a difference to the example of Figure 5.14, the navigation guard in Figure 5.15 can be only evaluated at instance level after query execution of the source analysis situation. For example, with respect to the instance of Figure 4.12, only after execution of source analysis situation `as1`, navigation guard `hasResult()` can be evaluated and only if it is true, the query of target analysis situation `as2` also is executed. Formally, the navigation step schema of 5.15 is written as $(AS1, \text{hasResult}(), \text{correlate}(\text{drillDownToLevel}(D, G), J), ((\text{Insurant}, \text{insDistrict}), \text{SameInsDistrict}), AS2)$ and the operator invocation at schema level as $AS2 = AS1.[\text{hasResult}()] \text{correlate}(\text{drillDownToLevel}(\text{Insurant}, \text{insDistrict}), \text{SameInsDistrict})$.

In the following subsection, we continue to present further examples of navigation step schemas regardless of whether navigation guards (unequal boolean expression `true`) are involved or not. Moreover, this subsection introduces examples of patterns that specify how navigation steps can be designed for specific analysis tasks.

5.3.3 Navigation Patterns

A navigation pattern can be considered as a reusable template that can be applied to a certain class of analysis problems to be solved. The notion of navigation pattern is used in a similar sense as notions like analysis patterns or design patterns. Navigation patterns are constructs at schema level that can make use of all constituents navigation step schemas can have. We do not introduce a list of navigation patterns but provide only several examples.

In the first examples we refer to specific navigation operators that are

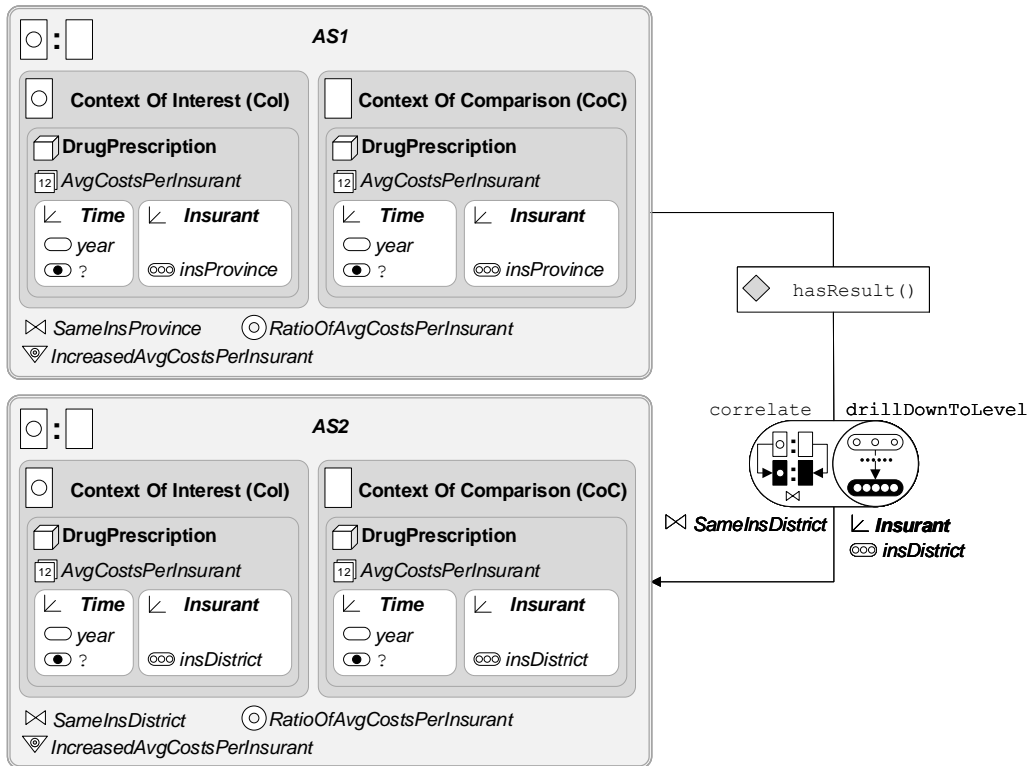


Figure 5.15: Example with navigation guard using operator `hasResult`

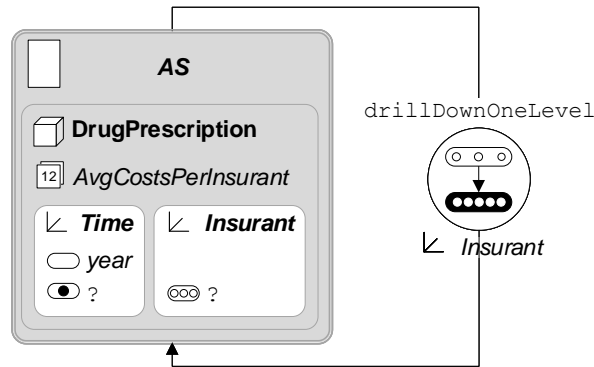


Figure 5.16: Navigation pattern to drill down to the next finer granularity level

predestined for specifying navigation patterns. Navigation operator `drillDownOneLevel` and navigation operator `rollUpOneLevel` change the gran-

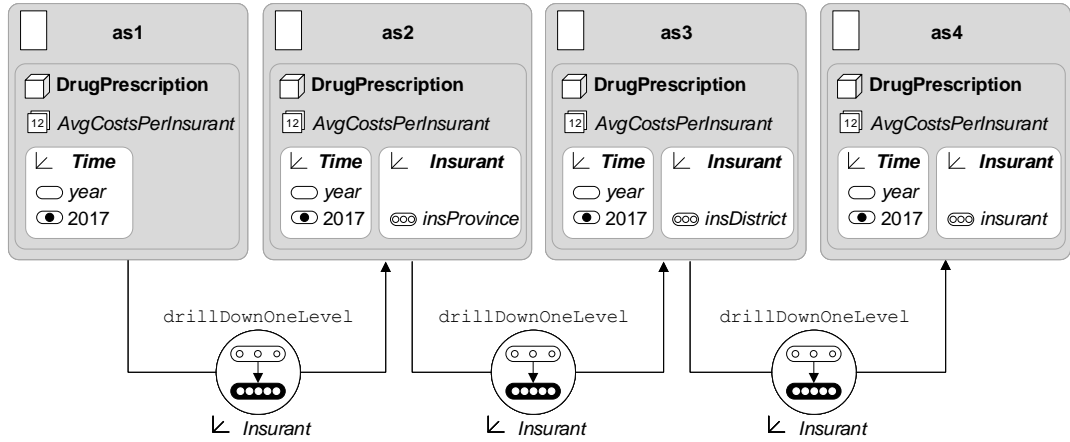


Figure 5.17: Instances generated by navigation pattern of Figure 5.16

ularity level of a dimension qualification to a finer or coarser one, respectively. Figure 5.16 shows a navigation step schema to drill down to the next finer granularity level in dimension qualification of dimension schema *Insurant*. Non-comparative analysis situation schema *AS* not only represents the source but also the target of the navigation step schema. Formally, the navigation step schema is written as $(AS, \text{true}, \text{drillDownOneLevel}(D), (Insurant), AS)$. Dice node $DiceNode_{AS}(Time)$ and granularity level $GranLvl_{AS}(Insurant)$ are specified as variables. Figure 5.17 demonstrates three instantiations of the navigation step schema presented in Figure 5.16. This instantiation process can be described following: First, variable $DiceNode_{AS}(Time)$ is bound to year 2017 and variable $GranLvl_{AS}(Insurant)$ to dimension level *top* yielding analysis situation *as1* with $DiceNode_{as1}(Time) = 2017$ and $GranLvl_{as1}(Insurant) = top$. The valid operator invocation $as1.drillDownOneLevel(Insurant)$ yields target analysis situation *as2* with $GranLvl_{as2}(Insurant) = insProvince$, i.e., $as2 = as1.drillDownOneLevel(Insurant)$. Target analysis situation *as2* can be taken again as an instance of source analysis situation schema *AS*, and the navigation step schema of Figure 5.16 can be applied a second time to obtain $as3 = as2.drillDownOneLevel(Insurant)$ and a third time to obtain $as4 = as3.drillDownOneLevel(Insurant)$. Finally, all instantiated navigation steps are depicted in

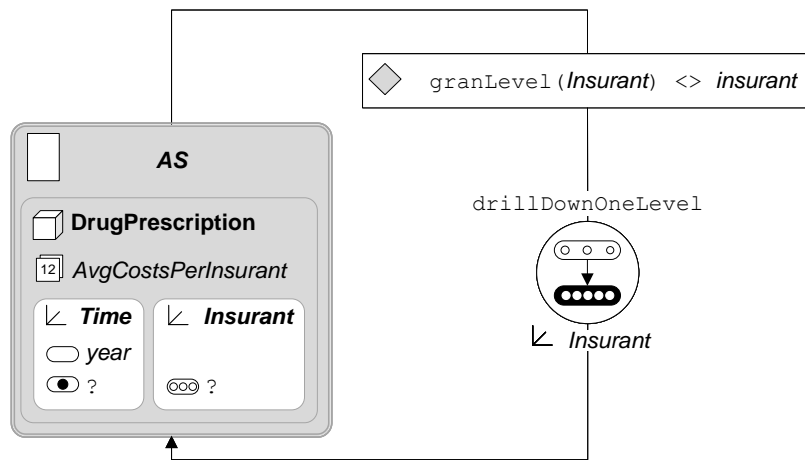


Figure 5.18: Navigation pattern of Figure 5.16 extended by a navigation guard to check the navigation operator's precondition

Figure 5.17.

The same pattern as demonstrated in Figure 5.16 is modeled in Figure 5.18 except that this navigation step schema comprises a navigation guard ($\text{granLevel}(\text{Insurant}) \neq \text{insurant}$) which is used to check the navigation operator's precondition (compare the definition of navigation operator `drillDownOneLevel` in Table 4.2 of Chapter 4). The navigation guard evaluates to true, if the granularity level of dimension schema *Insurant* of the source analysis situation schema is unequal to dimension level *insurant* that represents the base level of dimension schema *Insurant*. If the navigation guard evaluates to false, the granularity level of dimension schema *Insurant* of the source analysis situation schema is equal to dimension level *insurant* and, thus, no drill-down can be performed. But as demonstrated in Figure 5.17, the instantiation process of the navigation step schema is aborted, if the last drill-down operation leads to granularity level *insurant* because by dynamic type checking the navigation operator's precondition becomes false and no instantiation is performed. Hence, the modeling of navigation guard in Figure 5.18 is not necessary, it represents only an explicit information of the precondition at schema level.

Figure 5.19 represents a similar example as shown in Figure 5.16. The dif-

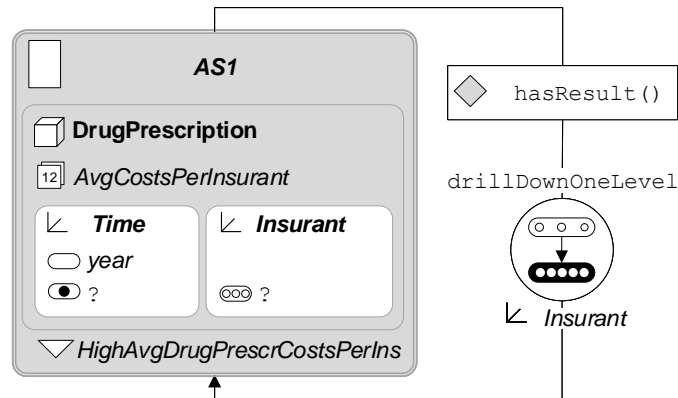


Figure 5.19: Navigation pattern to drill down to the next finer granularity level depending on the result set of the source

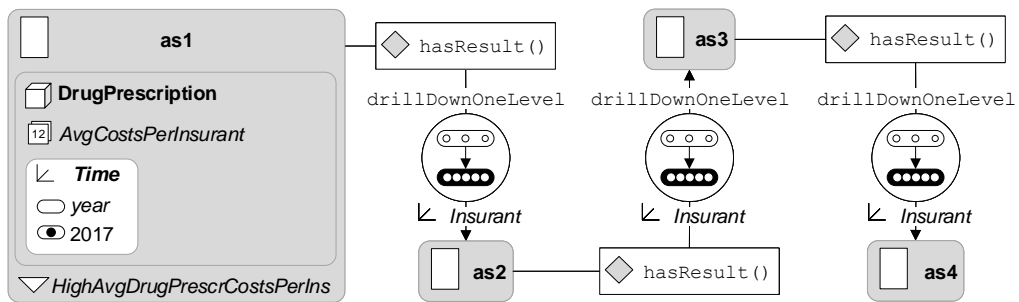


Figure 5.20: Instances generated by navigation pattern of Figure 5.19

ference is that analysis situation schema AS contains aggregate measure filter $HighAvgDrugPrescrCostsPerIns$ and the navigation step schema additionally comprise navigation guard $hasResult()$. An instantiated navigation step drills down to the next dimension level, but only, if the execution of the source analysis situation returns a non-empty result set. Semantically, this means that drilling down is only performed in the case of high average drug prescription costs per insurant. In Figure 5.20, three instances of navigation step schema of Figure 5.19 are depicted. Analysis situations $as2$, $as3$, and $as4$ are drawn in condensed graphical notation. As one sees, each instantiated navigation step comprises navigation guard $hasResult()$. Formally, the valid operator invocations are written as $as2 = as1.[hasResult()]drillDownOne-$

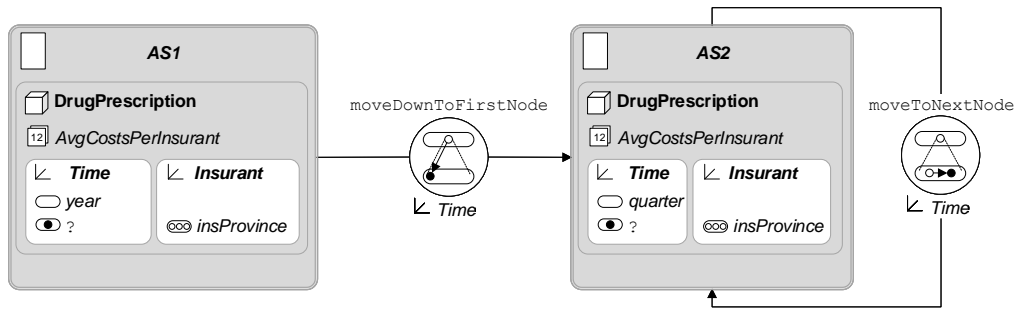


Figure 5.21: Navigation pattern to iterate over subnodes of a dimension node

$\text{Level}(\text{Insurant})$, $\text{as3} = \text{as2}.\text{[hasResult()]} \text{drillDownOneLevel}(\text{Insurant})$, and $\text{as4} = \text{as3}.\text{[hasResult()]} \text{drillDownOneLevel}(\text{Insurant})$.

Note, in both examples (first example in Figure 5.16 and Figure 5.17 and second example in Figure 5.19 and Figure 5.20), the precondition of navigation operator `drillDownOneLevel` must be satisfied for instantiation. Hence, for instance, operator call $\text{as4}.\text{[hasResult()]} \text{drillDownOneLevel}(\text{Insurant})$ does not represent a valid operator invocation because precondition $\text{GranLvl}_{\text{a4}}(\text{Insurant}) \neq \text{base}_{\text{Insurant}}$ (see Table 4.2 in Chapter 4) is false.²⁵ Therefore, it is not possible to instantiate a navigation step having analysis situation **a4** as source—this holds for both examples (first example demonstrated in Figures 5.16 and 5.17 and second example depicted in Figures 5.19 and 5.20).

In subsequent examples, we refer to specific navigation operators that can be used to iterate over subnodes of a dimension node. In Table 4.4 and Table 4.5 of Chapter 4, navigation operators are specified to move to the first or last subnode of a dimension node (`moveDownToFirstNode` and `moveDownToLastNode`), or to move to the next or previous subnode of a dimension node (`moveToNextNode` and `moveToPrevNode`). These operators can be used to iterate over all subnodes of a dimension node from the first to the last subnode or, inversely, from the last to first node with respect to a defined order relation of the dimension nodes of the corresponding dimension

²⁵The precondition evaluates to $\text{GranLvl}_{\text{a4}}(\text{Insurant}) = \text{insurant} \neq \text{insurant} = \text{base}_{\text{Insurant}}$ which is false.

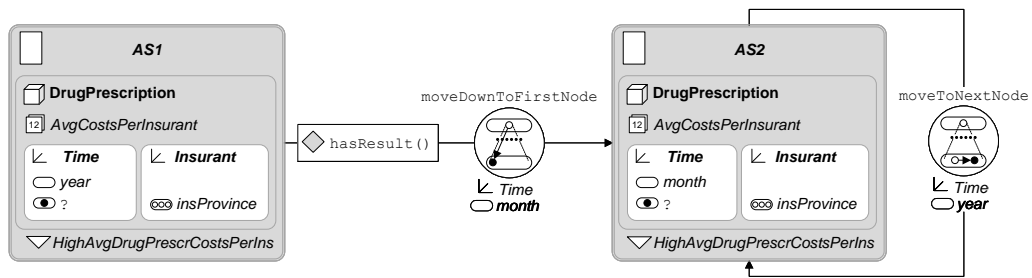


Figure 5.22: Navigation pattern to iterate over subnodes of a dimension node depending on the result set of the source

level. Two variants allow to iterate over direct subnodes or over subnodes with respect to an arbitrary sublevel.

Figure 5.21 shows two navigation step schemas. The first one comprises source analysis situation schema *AS1* and target analysis situation schema *AS2*, and the second navigation step schema has analysis situation schema *AS2* as source and target. In the dimension qualification of *AS1* concerning dimension schema *Time*, the dice node is specified as a variable and refers to dimension level *year*. The navigation step schema from analysis situation *AS1* to analysis situation *AS2* allows to instantiate navigation steps to navigate from year analysis to analysing the first quarter of the year—done by operator invocation (at schema level) $AS2 = AS1.moveToFirstNode(Time)$. In target analysis situation schema *AS2*, the dimension qualification referring dimension schema *Time* comprises a variable that can be bound to dimension nodes of dimension level *quarter*. Navigation step schema $(AS2, true, moveToNextNode(D), (Time), AS2)$ allows to iterate over all quarters of a year at instance level. Let *as1* be an instance of *AS1* where the year variable is bound to 2017, and let *as21*, *as22*, *as23*, and *as24* be analysis situations that are instances of *AS2* where the quarter variable is bound to 2017Q1, 2017Q2, 2017Q3, and 2017Q4, respectively, then the following navigation steps can be instantiated by the navigation step schemas of Figure 5.21: $(as1.moveToFirstNode(Time), as21, true)$, $(as21.moveToNextNode(Time), as22, true)$, $(as22.moveToNextNode(Time), as23, true)$, and $(as23.moveToNextNode(Time), as24, true)$.

Figure 5.22 presents an example similar to the example in Figure 5.21. One difference is that the navigation step schemas depicted in Figure 5.22 allow to iterate over the month of a year. Another difference is that this iteration is only done, if after execution of an instance of analysis situation schema *AS1*, a non-empty result set is obtained. This is evaluated by navigation guard `hasResult()`. In combination with aggregate measure filter *HighAvgDrugPrescrCostsPerIns*, this navigation pattern allows to evaluate the average drug prescription costs per insurant of each month of a year, if the average drug prescription costs per insurant of the year itself is high.

In Figure 5.23, a navigation pattern is depicted that iterates over dice nodes and drills down to a sublevel. By analysis situation schema *AS1*, two years (represented by a variable in the context of interest and by another variable in the context of comparison) can be compared with respect to average drug prescription costs per insurant. As a score, the ratio of the average costs per insurant is computed. Only those records are filtered that show a cost increase. The navigation step schema from analysis situation schema *AS1* to analysis situation schema *AS2* performs a correlated move down to the first insurants' province.²⁶ The comparative navigation step schema from analysis situation schema *AS2* to itself iterates over all provinces, i.e., year comparison is done for all insurants' provinces. For a province with cost increase, also its districts are listed for comparison. This is modeled by navigation step schema from *AS2* to *AS3* where navigation guard `hasResult()` controls navigation. If the result set of an instance of *AS2* is non-empty, then navigation to the corresponding instance of *AS3* is performed and the query of this instance is executed, otherwise, if the result set is empty, the navigation step is not performed and the query of the target is not executed.

Figure 5.24 represents the same navigation pattern as depicted in Figure 5.23 with the difference that analysis situation schemas *AS2* and *AS3* are shown in condensed graphical notation. Comparative analysis situation *AS1*

²⁶Note, in lean graphical notation of analysis situation schema *AS1*, the dimension qualification with respect to dimension schema *Insurant* is not depicted, although, formally it is present and contains dice node `all` and dice level `top`. Hence, navigation operator `moveDownToFirstNode` moves from dice node `all` of dice level `top` to the first province of dice level `province`.

is drawn in lean graphical notation. Note that also in Figure 5.24, the whole information of analysis situations *AS2* and *AS3* can be also constructed. The navigation operators show the difference between source and target. Thus, from a source analysis situation schema also the target analysis situation schema can be fully specified.

5.4 Discussion

In this chapter, a clear distinction between schema and instance level for all design constructs has been made. This design criteria was continuously developed in our previous work. In [90], we introduced multi-dimensional navigation modeling using BI analysis graphs and BI analysis graph templates to represent re-usable and recurrent parts of analyses. BI analysis graph templates contain free variables that are bound to concrete values at instantiation time. A distinction between generic and individual analysis situation, and between generic navigation steps and navigation steps was made in [91]. Generic analysis situations and generic navigation steps contain free variables and correspond to analysis situation schemas and navigation step schemas. Individual analysis situations and navigation steps correspond to instances.

Moreover, in [91], we proposed inheritance for generic analysis situations and generic navigation steps to increase re-usability. Generic analysis situations and generic navigation steps could be specialized to allow type-safe differentiation of various analysis situations and navigation steps with a common generic analysis situation schema and generic navigation step. In APMN4BI, we exchanged this approach by introducing navigation guards at schema level. Differentiation is done by navigating to different analysis situation schemas controlled by navigation guards. We choose this approach because business analysts have a better understanding for branching by navigation guards than for “branching by inheritance” (an insight gained from case studies).

Navigation guards are constructs that must be considered at two different levels. There are navigation guards to check the result set of a source ana-

lysis situation after query execution. This kind of navigation guards can be only applied at instance level and must be a component of a navigation step. The second type of navigation guards is applied on the components of source analysis situations to decide whether a navigation step can be performed or not. If a navigation step cannot be performed, this navigation step can be omitted at all. Hence, this kind of navigation guards represent navigation guards at schema level (nevertheless, they can be also considered as a part of navigation steps that are never performed). Thus, it is sufficient to examine navigation guards at schema level at instantiation time. For simplicity (to avoid case distinctions), in the formal definitions, we do not distinguish between navigation guards at schema level and navigation guards at instance level. Formally, a navigation guard of a navigation step schema also becomes a navigation guard of an instance of this navigation step schema. In graphical representations, after instantiation one can omit such navigation steps that contain navigation guards which are always evaluated to false.

In [91], guidance rules were introduced for additional navigation control (also in combination with inheritance). They are defined over analysis situations and provide recommendations on how to proceed in analysis depending on an analysis situation's query result. In APMN4BI, we do not define the explicit concept of guidance rules but navigation in APMN4BI can be additionally controlled by filter conditions (aggregate measure filters for non-comparative analysis situations and score filters for comparative analysis situations), and navigation guards `hasResult()` and `hasNoResult()`. The provision of aggregate measure filters and score filters is useful independently from navigation. In combination with navigation guards, guidance rules can be simulated in an understandable way. Therefore, an additional construct of guidance rules is not needed and an overload of APMN4BI by additional language constructs can be avoided. The notions of filter conditions and navigation guards are well understandable and sufficient for this type of navigation control.

Judgment rules and analysis rules represent further constructs represented in [91, 120]. The first one provide informative judgments and the second one actions both based on the result set of an analysis situation.

These constructs are parts of the presentation & action layer. Hence, they are out of scope of APMN4BI as presented in the current thesis. More about business rules comprising analysis, judgment, and guidance rules can be found in [111].

The notion of type compliance was introduced in this chapter and interrelated to the notion of type safety of programming languages. In the conceptual modeling language APMN4BI, type compliance can be interpreted as schema compliance. A navigation step is compliant with respect to a navigation step schema, if the source and target analysis situations of the navigation step are instances of the source and target analysis situation schemas of the navigation step schema, respectively, and the navigation operator of the navigation step schema also occurs in the navigation step with the same actual parameter list as it is specified in the navigation step schema with the exception that all free variables are bound to constants; moreover, it is necessary that the operator's pre- and postcondition (including the frame assumption) are satisfied for the navigation step. As defined in this chapter, such a schema compliant navigation step represents an instance of the corresponding navigation step schema. In this context, we also discussed the point in time when such type checking can be performed. If it is possible to check already at schema level that a navigation step schema only induces type compliant navigation steps, one can consider this as static type checking. Otherwise, if the checking whether a resulting navigation step is an instance of a navigation step schema is only possible after all free variables of that navigation step schema are bound, we speak of dynamic type checking. Both notions, static and dynamic type checking, are similar to the same notions in the context of programming languages where type checking can be performed at compile time or only at runtime.

Navigation step schemas are constructs of a conceptual modeling language. We do not investigate implementation perspectives like performance optimizations. For instance, the navigation pattern in Figure 5.23 presents an example that iterates over provinces and, in the case of a cost increase, a drill down operation to districts is performed. A translation into queries (at instance level) generates one query per province and another query to drill

down to districts of this province. This view is sufficient at a conceptual level and provides a view for business analysts. If one takes into account aspects of performance optimization, she or he could consider an implementation approach that performs a query which selects all provinces having a cost increase and, afterwards, iterates over the result set and performs queries that list districts of a province.

In the following chapter, we introduce business intelligence (BI) analysis graphs and business intelligence (BI) analysis graph schemas. Again, the concept is presented at instance and schema level. It finalizes the approach of APMN4BI and presents how to specify whole analysis processes.

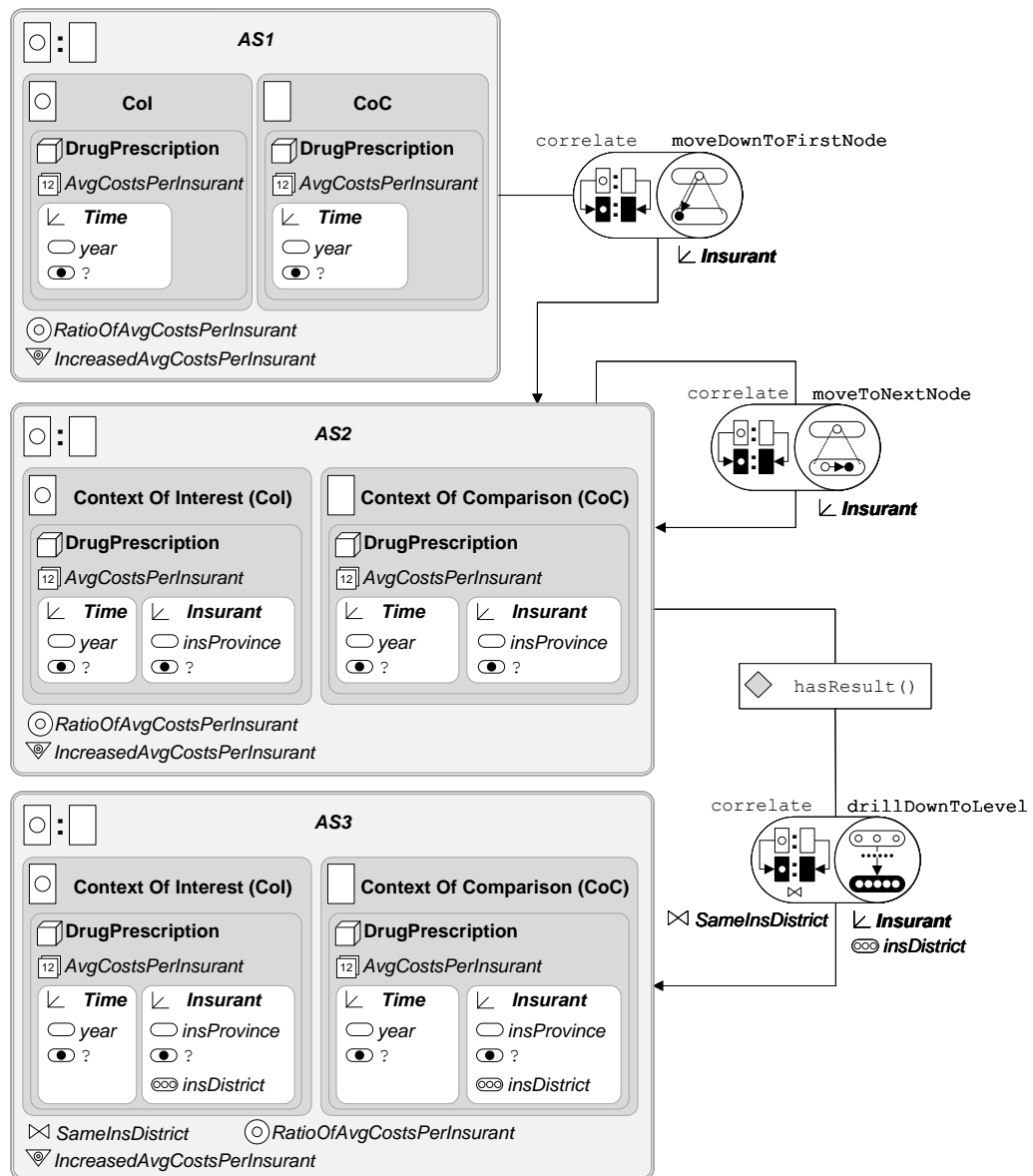


Figure 5.23: Navigation pattern iterating over subnodes and drilling down to sublevel depending on a navigation guard

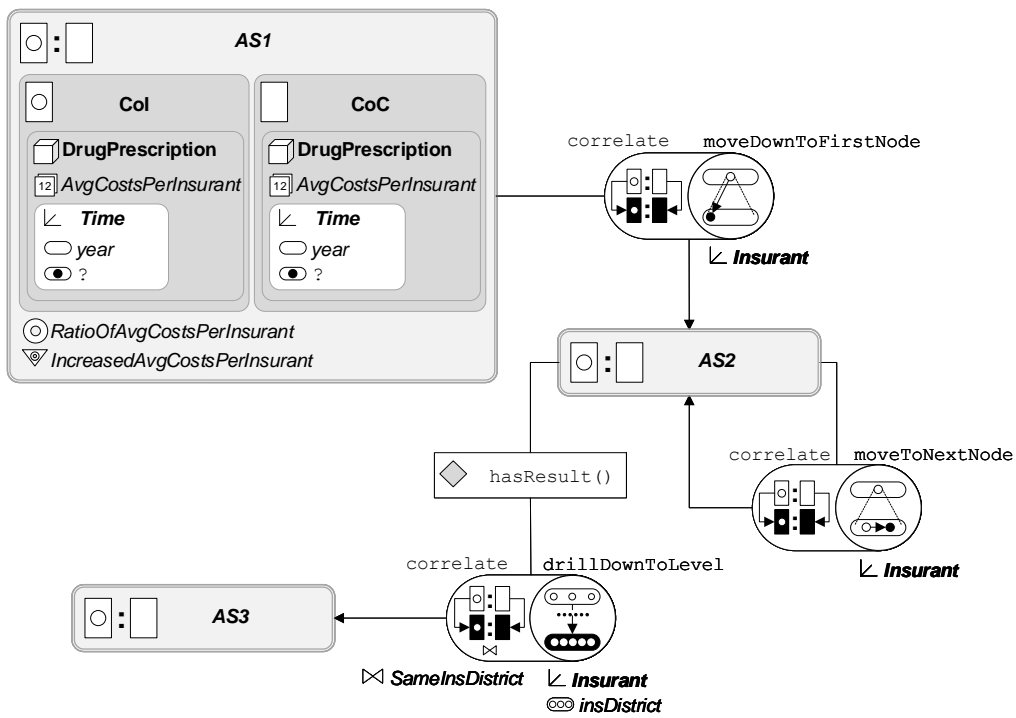


Figure 5.24: Example of Figure 5.23 presenting analysis situation schemas in condensed graphical notation

Chapter 6

Business Intelligence (BI) Analysis Graphs

Contents

6.1	Definition of BI Analysis Graphs	283
6.1.1	Named Analysis Situations	284
6.1.2	Formal Definition of BI Analysis Graphs	286
6.2	BI Analysis Graph Schemas	289
6.2.1	Named Analysis Situation Schemas	290
6.2.2	Formal Definition of BI Analysis Graph Schemas	292
6.3	Instances of Analysis Graph Schemas	302
6.4	Structuring by Subgraphs	311
6.5	Composite Analysis Situation	326
6.5.1	Preliminary Definitions	327
6.5.2	Formal Definition at Schema Level	331
6.5.3	Formal Definition at Instance Level	333
6.5.4	Graphical Representation and Examples	336
6.6	Analysis Trace and Backtracking	352
6.7	Discussion	358

Based on the previous chapters, this chapter presents the core of APMN4BI: business intelligence (BI) analysis graphs and BI analysis graph schemas. BI analysis graphs represent specific analysis processes which can also be considered as OLAP sessions. They comprise a set of named analysis situations (both non-comparative and comparative) linked by navigation steps both without unbound variables. A BI analysis graph can be used to document a specific analysis process to increase comprehensibility. A business analyst specifies a start analysis situation that represents the first query which is executed to obtain the first result set. Based on the start analysis situation, the analysis process can be stopped or continued. If it is continued, one or more analysis situations are derived from the start analysis situation by invoking navigation operators. The navigation operator and its actual parameters make the difference of both linked analysis situations visible. Each generated target analysis situation can be used to derive another analysis situation by applying again navigation operators, i.e., a target analysis situation becomes the source analysis situation of another navigation step. Such an analysis process depicted by a BI analysis graph represents a directed tree.

APMN4BI models represent BI analysis graph schemas. A BI analysis graph cannot only be considered as a recorded analysis process but it could also be a result of a proactively modeled analysis process that can be executed. To increase re-usability, one can add analysis situations and navigation steps with unbound variables and to provide process control, navigation guards can be used. APMN4BI models are BI analysis graph schemas that are modeled proactively and that can be instantiated yielding a specific BI analysis graph, i.e., yielding a specific analysis process. One can say, the execution of a BI analysis graph schema creates a BI analysis graph which represents a specific OLAP session. During this execution, free variables have to be bound and navigation guards have to be evaluated. As one can see later, a BI analysis graph schema can be considered as a directed multi-graph having edges with own identity. The consideration as a multi-graph permits at schema level to have multiple navigation step schemas from the

same source and to the same target analysis situation schema.

The first section of this chapter starts with the definition of BI analysis graphs. In the second section, we introduce BI analysis graph schemas. The instantiation of BI analysis graph schemas is described in the third section. To preserve an overview in comprehensive analysis processes, subgraphs are used to structure BI analysis graphs and BI analysis graph schemas in a hierarchical way. This structuring by subgraphs is presented in the fourth section of this chapter. Subgraphs themselves represent BI analysis graphs or BI analysis graph schemas which are embedded in other BI analysis graphs or a BI analysis graph schemas, respectively. Composite analysis situations and composite analysis situation schemas are introduced in another section. They can be considered as specific subgraphs at instance and at schema level, respectively. A composite analysis situation schema is instantiated as a whole and in one go in a unique and automatic way without additional user interactions (except for the initiation of the instantiation process). Another section in this chapter refers to the temporal order of executing BI analysis graphs (analysis trace) including backtracking steps to jump back to previous analysis situations for tracking alternative branches of a BI analysis graph. Finally, this chapter is concluded by a section that again emphasizes the distinction between schema and instance level and the purpose of this distinction with respect to modeling and execution of analysis processes. This section also discusses previous work and approaches that already contain incipient stages of the separation between schema and instance level.

6.1 Definition of BI Analysis Graphs

A BI analysis graph represents a specific analysis process that has a start analysis situation and several branches of navigation steps. Formally, a BI analysis graph can be defined as a directed tree that has analysis situations as vertices and navigation steps as edges. The start analysis situation represents the root node. In this section, first we introduce preliminary definitions of named analysis situations and navigation steps comprising named analysis situations. Afterwards, we provide the formal definition of BI analysis graphs.

6.1.1 Named Analysis Situations

In Chapter 3, we introduced non-comparative and comparative analysis situations. The formal definitions do not contain unique identifiers for analysis situations. Two analysis situations are equal, if their defining components are equal. This is also compatible with respect to the semantics of analysis situations. Equal analysis situations induce equal queries. In the previous chapters, we only used names for analysis situations at a meta level and in graphical representations.

During an analysis process, it is also possible that a navigation step generates an analysis situation that already exists at another position of the tree (representing a BI analysis graph), i.e., the same analysis situation can occur at different vertices. To overcome this problem, we introduce named analysis situations that formally provide a unique identifier. The name of named analysis situations represents a defining component—it is not a name at a meta level anymore. As a consequence the notion of navigation steps also has to be extended to navigation steps with named analysis situations.

Definition 6.1. A *named analysis situation* $\mathbf{nas} = (id, \mathbf{as})$ comprises

- an identifier id (the name of \mathbf{nas}) and
- a non-comparative or comparative analysis situation \mathbf{as} .

Furthermore, in the context of \mathbf{nas} , we define $id_{\mathbf{nas}} = id$ and $as_{\mathbf{nas}} = \mathbf{as}$. \square

Named analysis situations are used as vertices in the definition of BI analysis graphs. Later, in the formal definition of BI analysis graphs, we will see that, at least within one analysis graph, these identifiers have to be unique. $id_{\mathbf{nas}}$ and $as_{\mathbf{nas}}$ represent the identifier (the name) and the analysis situation itself of a named analysis situation \mathbf{nas} .¹ In the graphical presentation, we already used names for analysis situations. Thus the graphical presentation need not be adapted.

¹In this thesis, we also use $id_{\mathbf{nas}}$ and \mathbf{nas} equivalently to refer to named analysis situation \mathbf{nas} . Furthermore, if the context is clear, we also write “analysis situation” instead of “named analysis situation”.

At this point, we provide an additional remark concerning derived cubes introduced in Chapter 5. Derived cubes are obtained from non-comparative analysis situations and can be used as a cube in another analysis situation. A derived cube is based on the query of a non-comparative analysis situation. This query can be re-used in a view definition and the name of the view serves as a name of the derived cube. The extension to named analysis situations allows to obtain names for such view definitions in a systematic way.

Finally, we have to introduce another auxiliary definition that extends a navigation step to a navigation step with named analysis situations. Together with named analysis situations, this extension provides compatibility in the definition of BI analysis graphs. Navigation steps with named analysis situations represent edges of the directed tree that specifies a BI analysis graph.

Definition 6.2. A *navigation step with named analysis situations* $\mathbf{nav} = (\mathbf{src}, \mathbf{navStep}, \mathbf{trg})$ comprises

- a named analysis situation \mathbf{src} (*named source analysis situation*),
- a named analysis situation \mathbf{trg} (*named target analysis situation*) with $id_{\mathbf{src}} \neq id_{\mathbf{trg}}$, and
- a navigation step $\mathbf{navStep}$ with $Source_{\mathbf{navStep}} = as_{\mathbf{src}}$ and $Target_{\mathbf{navStep}} = as_{\mathbf{trg}}$.

Moreover, in the context of \mathbf{nav} , we define $Source_{\mathbf{nav}} = \mathbf{src}$, $Target_{\mathbf{nav}} = \mathbf{trg}$, $SourceId_{\mathbf{nav}} = id_{\mathbf{src}}$, $TargetId_{\mathbf{nav}} = id_{\mathbf{trg}}$, $SourceAS_{\mathbf{nav}} = as_{\mathbf{src}}$, $TargetAS_{\mathbf{nav}} = as_{\mathbf{trg}}$, and $NavStep_{\mathbf{nav}} = \mathbf{navStep}$. \square

This definition of a navigation step with named analysis situations \mathbf{nav} is based on a navigation step $NavStep_{\mathbf{nav}}$ accordingly to Definition 4.3 of Chapter 4. A navigation step with named analysis situations represents an edge of a BI analysis graph. To ensure uniqueness of the source and target vertices, it is required that $SourceId_{\mathbf{nav}}$ and $TargetId_{\mathbf{nav}}$ are different. If the context is clear, for convenience, we also simply say navigation step instead of navigation step with named analysis situations. After these auxiliary

definitions, in the following subsection, we continue with the formal definition of BI analysis graphs.

6.1.2 Formal Definition of BI Analysis Graphs

A BI analysis graph can be considered as a directed tree which reflects an analysis process. More specifically, one can also think of OLAP sessions. The vertices of a BI analysis graph are named analysis situations and the directed edges are navigation steps with named analysis situations as source and target vertices. The subsequent definition formalizes this idea.

Definition 6.3. A *business intelligence (BI) analysis graph* $\mathbf{ag} = (V, E)$ is a directed tree that comprises

1. a set V of named analysis situations as vertices and
2. a set E of directed edges representing navigation steps with named source and target analysis situations.

Moreover, in the context of \mathbf{ag} , we define $ASituations_{\mathbf{ag}} = V$ and $NavSteps_{\mathbf{ag}} = E$. A navigation step $\mathbf{nav} \in NavSteps_{\mathbf{ag}}$ is also referred as $SourceId_{\mathbf{nav}} \rightarrow TargetId_{\mathbf{nav}}$. The root of the directed tree \mathbf{ag} is denoted as $Root_{\mathbf{ag}}$. \square

A BI analysis graph has named analysis situations as vertices. Thus, each vertex represents a concrete query on an eDFM and each navigation step uniquely leads to another vertex that represents another named analysis situation. Navigation steps are the edges of a BI analysis graph. Because at least the identifiers of the named source and target analysis situations of a navigation step have to be different (accordingly to Definition 6.2), all vertices of a BI analysis graph also have to be different, i.e., for a BI analysis graph \mathbf{ag} , if $\mathbf{nas}_1, \mathbf{nas}_2 \in ASituations_{\mathbf{ag}}$ with $\mathbf{nas}_1 \neq \mathbf{nas}_2$, then $id_{\mathbf{nas}_1} \neq id_{\mathbf{nas}_2}$. But note that, in this case, it is allowed that $as_{\mathbf{nas}_1} = as_{\mathbf{nas}_2}$ meaning that the underlying analysis situations (and, respectively, the underlying queries) may be equal. Finally, note that an edge (a navigation step) of a BI analysis graph also contains navigation guards, although, in the case a navigation guard is defined by boolean expression `true`, one can think casually of a “missing”

navigation guard. Thus, in graphical representations of BI analysis graphs, we do not draw navigation guards of boolean expression `true`.

Note, the definition as a directed tree also allows exceptional cases where the set of edges, or even where both the set of vertices and the set of edges are empty. Whereas the case with empty sets for both can be considered as a theoretical construct, the case where $ASituations_{\mathbf{ag}} \neq \emptyset$ and $NavSteps_{\mathbf{ag}} = \emptyset$ represents a BI analysis graph \mathbf{ag} with a single analysis situation (the root) but without a navigation step. Because a tree must have connected vertices, in this case, $ASituations_{\mathbf{ag}}$ can only contain one single element.

Figure 6.1 demonstrates an example of a BI analysis graph. It is based on the running example introduced in Section 2.1 (compare with Figure 2.1). The underlying eDFM is based on the example presented in Section 2.4 (see Figure 2.3). In subsequent presentations, we also use the identifier (the name) of a named analysis situation for referring the named analysis situation itself and also for referring the contained analysis situation.

The first analysis situation of Figure 6.1 with identifier `as-1` represents the root of the directed tree. It selects drug prescription costs of year 2016 per insurants' province. There is one edge from `as-1` to `as-2` that introduces comparison (navigation step `as-1` \rightarrow `as-2`). In this navigation step, analysis situation `as-1` is transferred as context of interest to analysis situation `as-2`. On the other side, operator `moveToPrevNode` is invoked for dimension schema *Time* on analysis situation `as-1`. The result is transferred as context of comparison by operator `relate`. Analysis situation `as-2` compares drug prescription costs of 2016 with drug prescription costs of 2015. Comparison is done per insurants' province. Provinces of the context of interest and provinces of the context of comparison are joined by condition *SameInsProvince*. The ratio of both drug prescription costs is returned as score *RatioOfSumOfCosts*. Note, in this example, there is only one edge going out from root `as-1`.

From analysis situation `as-2`, there is also only one edge going out to analysis situation `as-3` (navigation step `as-2` \rightarrow `as-3`). In this navigation step, a score predicate is added to the set of score filters of `as-3`. In analysis situation `as-3`, only result rows with significant costs increase are returned. Sig-

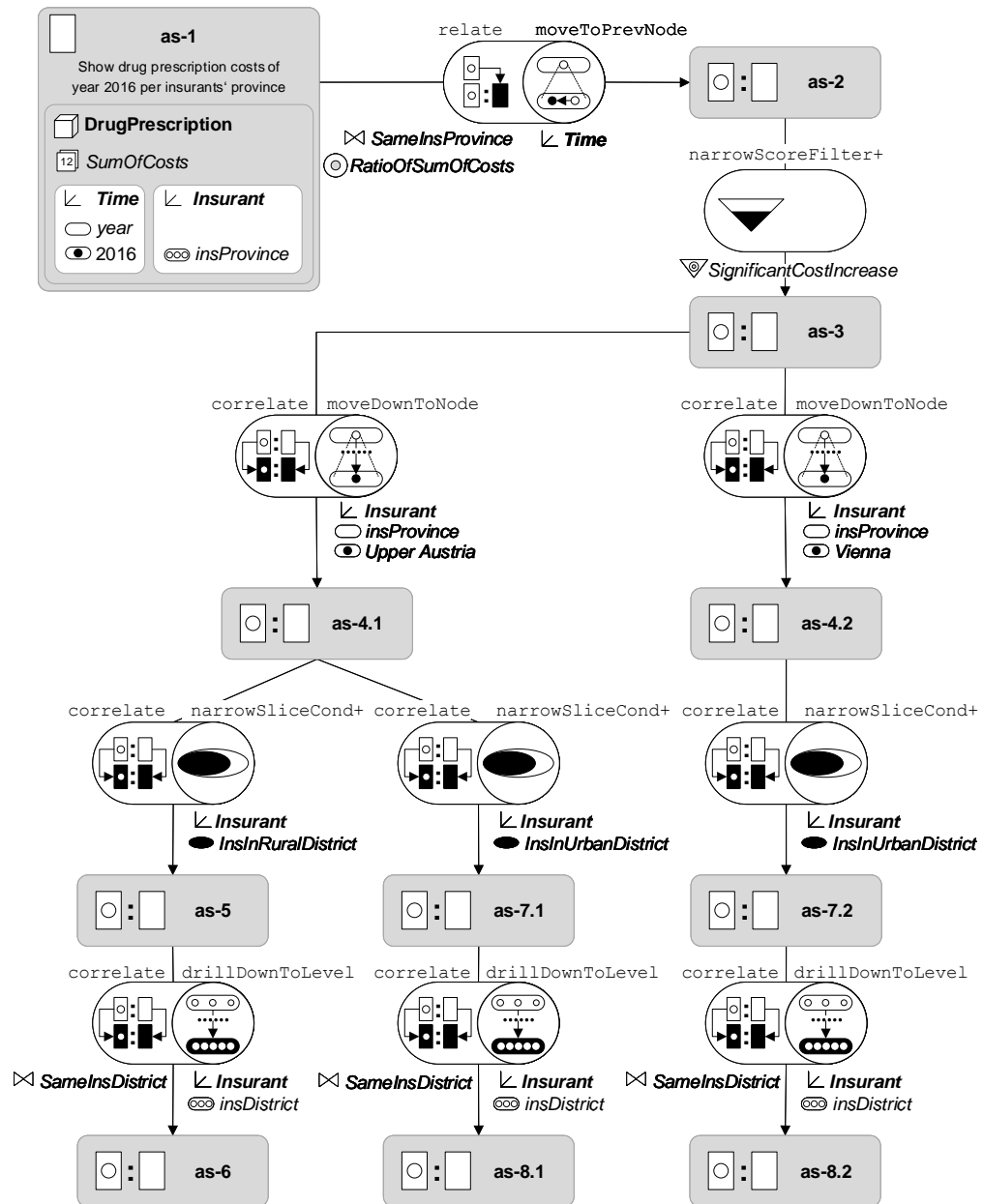


Figure 6.1: Example of a BI analysis graph extracted from the running example presented in Section 2.1

nificant cost increase is specified by score predicate *SignificantCostIncrease*. In the eDFM presented in Figure 2.3 of Chapter 2, score predicate *SignificantCostIncrease* is not depicted. Boolean expression *RatioOfSumOfCosts > 1.05* represents a possible definition of this score predicate.

There are two branches that have analysis situation *as-3* as source. One branch moves down correlated to node *Upper Austria* and the other one to *Vienna*, both Austrian provinces. The first of these navigation steps has as target *as-4.1* and the second *as-4.2*. Starting from *as-4.1*, there are two branches, one with navigation steps to analysis situations *as-5* and *as-6*, and the second branch to analysis situations *as-7.1* and *as-8.1*. Navigation step *as-4.1* → *as-5* narrows to insurants of rural districts and navigation step *as-5* → *as-6* lists result rows per district.² Similarly, in the second branch, analysis situation *as-7.1* is restricted to insurants of urban districts and, again, result rows are listed per district in analysis situation *as-8.1*. The navigation steps *as-4.2* → *as-7.2* and *as-7.2* → *as-8.2* invoke the same operators except not for *Upper Austria* but for *Vienna*. Also year comparison of drug prescription costs of urban districts listed per district is returned. Note, for *Vienna*, it makes no sense to analyze also drug prescription costs of rural districts. Thus, a second branch going out from *as-4.2* is missing.

In Figure 6.1, a BI analysis graph is shown that represents a specific analysis process. One could specify other BI analysis graphs representing other specific analysis processes but which are similar to that one in Figure 6.1. For example, one could add an additional branch for analyzing *Lower Austria*, again for rural and urban districts. In the subsequent section, BI analysis graph schemas are introduced that provide patterns which allow to instantiate specific analysis processes.

6.2 BI Analysis Graph Schemas

BI analysis graph of Figure 6.1 can be generalized in the sense that the year of interest and the selection of provinces are specified by unbound variables.

²Note, in this case of a drill-down operation also the join condition has to be re-defined. Therefore, the correlated drill-down also changes the join condition to *SameInsDistrict*.

Comparison of drug prescription costs for a province is done for rural and urban districts, except for Vienna. Such a process control can be modeled at schemas level by navigation guards. These and similar motivations toward generalization yields to the definition of BI analysis graph schemas to increase re-usability. A BI analysis graph schema defines a pattern to instantiate specific analysis processes or, in other words, it provides guidance for instantiating useful analysis situations for solving a specific category of analysis tasks.

Analogously to named analysis situations, in the first subsection, we introduce named analysis situation schemas and navigation step schemas that contain named analysis situation schemas. Finally, the second subsection actually provides the formal definition of BI analysis graph schemas. We will see that a BI analysis graph schema can be defined as a connected directed multi-graph comprising named analysis situation schemas as vertices and navigation step schemas containing named analysis situation schemas as edges.

6.2.1 Named Analysis Situation Schemas

Similarly to BI analysis graphs, also for BI analysis graph schemas auxiliary definitions of named analysis situation schemas and navigation step schemas containing named analysis situation schemas have to be provided. Analogously to analysis situations, the formal definitions of non-comparative and comparative analysis situation schemas introduced in Chapter 5 do not contain unique identifiers for analysis situation schemas. In the previous chapter, names for analysis situation schemas were only used at a meta level and in graphical representations.

BI analysis graph schemas are defined as connected directed multi-graphs and we allow that an analysis situation schema may occur at two different positions (vertices) in such a multi-graph. To obtain unique vertices, we use named analysis situation schemas. The name of a named analysis situation is not a name at a meta level anymore but it represents a defining component of a vertex. As a consequence, we also have to extend navigation step schemas

to navigation step schemas with named analysis situation schemas.

Definition 6.4. A *named analysis situation schema* $NAS = (ID, AS)$ comprises

- an identifier ID (the name of NAS) and
- a non-comparative or comparative analysis situation schema AS .

Furthermore, in the context of NAS , we define $ID_{NAS} = ID$ and $AS_{NAS} = AS$. □

Named analysis situation schemas are needed as vertices of a BI analysis graph schema. The same analysis situation schema can occur more than once. The vertices are distinguished by unique identifiers. ID_{NAS} returns the identifier (the name) of a named analysis situation schema and AS_{NAS} returns the “unnamed” analysis situation schema. The graphical representation needs no further adaptations because the graphical constructs of analysis situation schemas already have names.

If the context is clear, we allow to omit the word “named” in “named analysis situation schema”. Moreover, in the remainder of this thesis, we also use the identifier ID_{NAS} as a synonym for NAS . This means that we use the phrase “analysis situation schema ID_{NAS} ” equivalently to the phrase “analysis situation schema NAS ” and also equivalently to the phrase “analysis situation schema NAS with name (identifier) ID_{NAS} ”.

To provide compatibility, navigation step schemas are enhanced to navigation step schemas with named analysis situations schemas. Navigation step schemas with named analysis situations schemas are used as directed edges of a BI analysis graph schema.

Definition 6.5. A *navigation step schema with named analysis situation schemas* $NAV = (SRC, NavStepSchema, TRG)$ comprises

- a named analysis situation schema SRC (*named source analysis situation schema*),

- a named analysis situation schema TRG (*named target analysis situation schema*), and
- a navigation step schema $NavStepSchema$ with $Source_{NavStepSchema} = AS_{SRC}$ and $Target_{NavStepSchema} = AS_{TRG}$.

Moreover, in the context of NAV , we define $Source_{NAV} = SRC$, $Target_{NAV} = TRG$, $SourceId_{NAV} = ID_{SRC}$, $TargetId_{NAV} = ID_{TRG}$, $SourceASSchema_{NAV} = AS_{SRC}$, $TargetASSchema_{NAV} = AS_{TRG}$, and $NavStepSchema_{NAV} = NavStepSchema$. \square

A navigation step schema with named analysis situation schemas NAV is based on a navigation step schema $NavStepSchema_{NAV}$ as introduced in Definition 5.6 of Chapter 5. It represents an edge of a BI analysis graph schema. Similar to navigation steps with named analysis situations, if the context is clear, we also simply write (for convenience) navigation step schema instead of navigation step schema with named analysis situation schemas. Note, we do not require in the definition that $ID_{SRC} \neq ID_{TRG}$. Hence, we also allow to have loops as edges, i.e., we allow to have navigation step schemas where the same named analysis situation schema serves as source and target. In the following subsection, we continue with the formal definition of BI analysis graph schemas.

6.2.2 Formal Definition of BI Analysis Graph Schemas

BI analysis graph schemas are defined as directed multi-graphs with edges having their own identity. Named analysis situation schemas are vertices and navigation step schemas between named analysis situation schemas are edges. First we give a general definition of directed multi-graphs with edges having their own identity.

Definition 6.6. A *directed multi-graph with edges having their own identity* is defined as a 4-tuple (V, E, s, t) comprising

- a set of vertices V ,

- a set of edges E ,
- a function $s: E \rightarrow V$ (the *source mapping*), and
- a function $t: E \rightarrow V$ (the *target mapping*). □

We assume to have directed edges with their own identity and the functions s and t return the start and target vertex, respectively. Named analysis situation schemas represent vertices and navigation step schemas directed edges. It is also allowed to have multiple navigation step schemas between two vertices and it is also possible that a navigation step schema comprises a named analysis situation schema as source and target analysis situation schema (considered as a loop). This characteristics give rise to consider BI analysis graph schemas as multi-graphs with edges having their own identity.

Definition 6.7. A *business intelligence (BI) analysis graph schema* $AG = (V, E, s, t)$ is a connected directed multi-graph having edges with its own identity such that

1. the set of vertices V consists of named analysis situation schemas,
2. the set of directed edges E comprises navigation step schemas with named analysis situation schemas,
3. function $s: E \rightarrow V$ maps a navigation step schema $NAV \in E$ to its named source analysis situation schema $Source_{NAV}$, and
4. function $t: E \rightarrow V$ maps a navigation step schema $NAV \in E$ to its named target analysis situation schema $Target_{NAV}$.

Moreover, in the context of AG , we define $ASSchemas_{AG} = V$ and $NavSchemas_{AG} = E$. If $SRC, TRG \in ASSchemas_{AG}$, then $ID_{SRC} \Rightarrow ID_{TRG}$ denotes the set of navigation step schemas from SRC to TRG . □

Whereas a BI analysis graph can be defined as a directed tree that documents a specific analysis process, a BI analysis graph schema must be represented as a directed multi-graph having edges with own identity. We use

the notion of multi-graph to allow more than one navigation step schema between source and target. Each navigation step schema between two named analysis situation schemas has its own identity represented by the navigation operator and its parameters. Of course, a navigation step schema is directed from source to target leading to a directed edge.

As we have multi-graphs, we define $ID_{SRC} \Rightarrow ID_{TRG}$ as a set of navigation step schemas (a difference to navigation steps with notion \rightarrow). Furthermore, set $ID_{SRC} \Rightarrow ID_{TRG}$ only contains navigation step schemas in one direction, i.e., set $ID_{SRC} \Rightarrow ID_{TRG}$ contains navigation step schemas from named analysis situation schema SRC to named analysis situation schema TRG whereas $ID_{TRG} \Rightarrow ID_{SRC}$ comprises navigation step schemas from TRG to SRC . In this case, TRG represents the source and SRC the target.³ Set $ID_{SRC} \Rightarrow ID_{TRG}$ can be empty (there is no navigation step schema from SRC to TRG), can consist of exactly one navigation step schema from SRC to TRG , or it can contain multiple navigation step schemas from SRC to TRG .

Note, a BI analysis graph is a directed tree which is a connected graph per definition. For BI analysis graph schemas we have to claim the property of a connected graph explicitly. All analysis situation schemas have to be connected pairwise (directly or indirectly by one or more navigation step schemas). If this is not the case, we have more than one BI analysis graph schema.

Furthermore, the definition of BI analysis graph schemas also allows that $NavSchemas_{AG} = \emptyset$. In this case, analysis situations are only described by one analysis situation schema without a specification for navigation. Because a BI analysis graph schema must be a connected graph, in this case of missing navigation step schemas, $ASSchemas_{AG}$ can only contain at most one element. If both $ASSchemas_{AG}$ and $NavSchemas_{AG}$ are empty, one obtains the theoretical construct of an empty graph.

Figure 6.2 shows a BI analysis graph schema based on the eDFM of the running example introduced in Chapter 2. It also represents a generalization of the specific BI analysis graph of Figure 6.1. The first generalization

³Note that SRC and TRG are just names. The target of a navigation step schema can be a source of another navigation step schema.

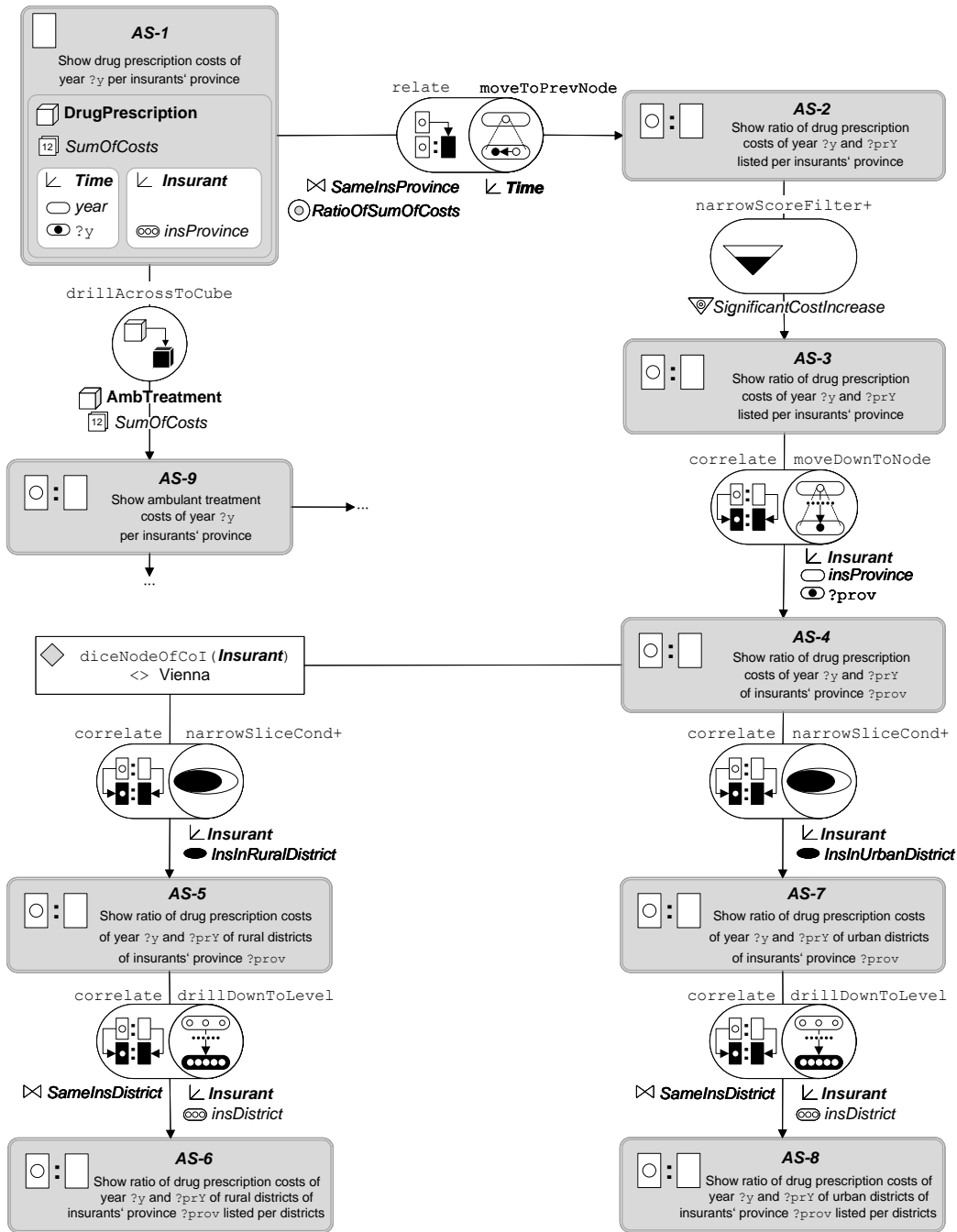


Figure 6.2: Example of a BI analysis graph schema extracted from the running example

concerns the analysis year. Analysis situation schema *AS-1* has an unbound variable $?y$ for dimension nodes in the dimension qualification of dimension schema *Time* at level *year*. Analysis situations of analysis situation schema *AS-1* show drug prescription costs of year $?y$ per insurants' province.

Set $AS-1 \Rightarrow AS-2$ contains one navigation step schema that transfers constituents of *AS-1* to the context of interest of *AS-2*. As context of comparison an analysis situation of schema *AS-2* obtains values derived from the source by applying navigation operator `moveToPrevNode` in dimension schema *Time*, i.e., year $?y$ is compared with previous year $?prY$.⁴ Note, although analysis situation schema *AS-2* is drawn in condensed graphical representation, one can conclude that *AS-2* must have another unbound variable $?prY$ beside unbound variable $?y$ (variable $?y$ for the dimension node in dimension schema *Time* of the context of interest and variable $?prY$ for the dimension node in dimension schema *Time* of the context of comparison). This consequence can be derived from the source, from the definition of operators `relate` and `moveToPrevNode`, and from the actual parameters. Both contexts are joined via insurants' provinces (join condition *SameInsProvince*), and `RatioOfSumOfCosts` is used as score.

Analysis situation schemas *AS-2* and *AS-3* are source and target of navigation step schema with operator `narrowScoreFilter+` that adds score filter *SignificantCostIncrease*. Analysis situations of comparative analysis situation schema *AS-2* only list rows with significant cost increase.

The single navigation step schema in set $AS3 \Rightarrow AS4$ specifies a correlated move down to a province in dimension schema *Insurant*. The province is specified by unbound parameter $?prov$ of operator `moveDownToNode`, i.e., at instantiation time of this navigation step this unbound parameter has to be bound to a specific province (for example, to *Upper Austria*, to *Lower Austria*, or to *Vienna*). Note that due to the given navigation operation, we can conclude that also the dice nodes of both context of interest and context of comparison have to be unbound variables. Operator `correlate` assures that both unbound variables are bound to the same value, i.e., to the same

⁴Note that we also use variable names in the descriptions of analysis situation schemas for better readability.

province. We assume that the unbound variable of one context is named by `?prov` such that it can be used in the description of *AS-4*.

Analysis situation schema *AS-4* is a source of two navigation step schemas. In the single navigation step schema of $AS-4 \Rightarrow AS-5$, we have a navigation guard that examines the dice node of the context of interest (given by an unbound variable). The navigation step may only be invoked in the case of provinces having rural districts. Thus the navigation guard has to exclude province Vienna. Analysis situations of schema *AS-5* compare drug prescription costs of a specific province restricted to rural districts. The navigation step schema in set $AS-5 \Rightarrow AS-6$ specifies a drill down to level *insDistrict* in dimension schema *Insurant*. The comparison results are listed per rural district.

The second branch going out of *AS-4* specifies a similar navigation step schema as in the first branch. Navigation step schema from *AS-4* to *AS-7* narrows to urban districts and the navigation step schema in $AS-7 \Rightarrow AS-8$ lists result rows per insurants' district. There is no navigation guard between *AS-4* and *AS-7* because each Austrian province has urban districts.

The BI analysis graph schema of Figure 6.2 can be extended to cost comparison of ambulant treatment and hospitalization. Thus, there can be defined further navigation step schemas that can be used to create navigation steps that navigate to cubes of other cube schemas. Such possible extensions are indicated by further navigation step schemas like the one from analysis situation schema *AS-1* to analysis situation schema *AS-9*.

In Figure 6.3, the example of Figure 6.2 was modified. There are two navigation step schemas from source analysis situation schema *AS-3* to target analysis situation schema *AS-4* in Figure 6.3 instead of one in Figure 6.2. Whereas in Figure 6.2, the cardinality of set $AS3 \Rightarrow AS4$ has value $|AS3 \Rightarrow AS4| = 1$, the cardinality of $AS3 \Rightarrow AS4$ in Figure 6.3 is $|AS3 \Rightarrow AS4| = 2$. Figure 6.3 presents an example that requires the definition of a multi-graph to provide multiple edges between two vertices. Finally, in Figure 6.3, the navigation guard can be omitted. The navigation step schemas from *AS-3* to *AS-4* restrict navigation to **Upper Austria** and **Lower Austria**. Both are rural provinces such that the navigation guard of Figure 6.2 is not needed

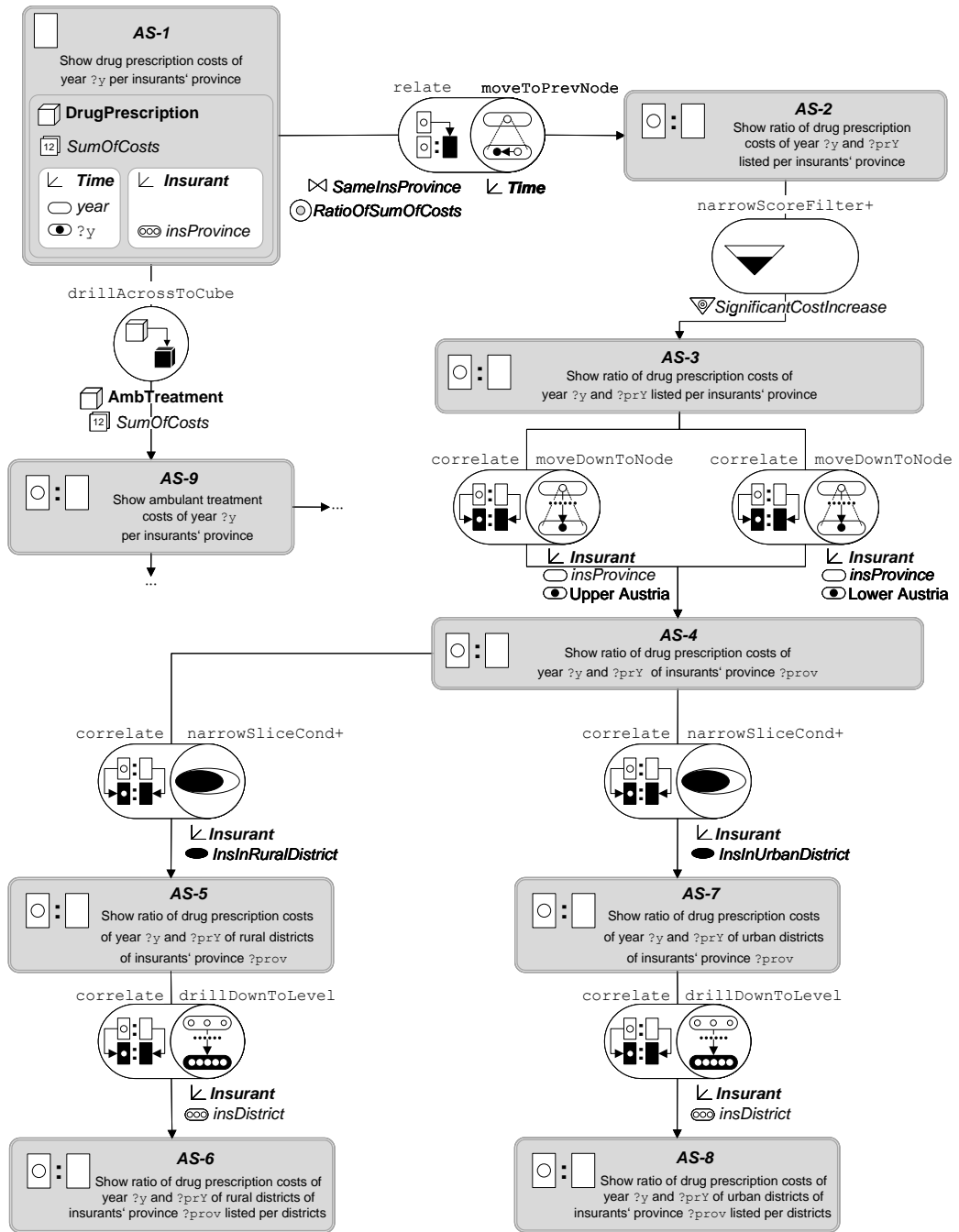


Figure 6.3: Modified BI analysis graph schema of Figure 6.2 containing more than one navigation step schema between two analysis situation schemas

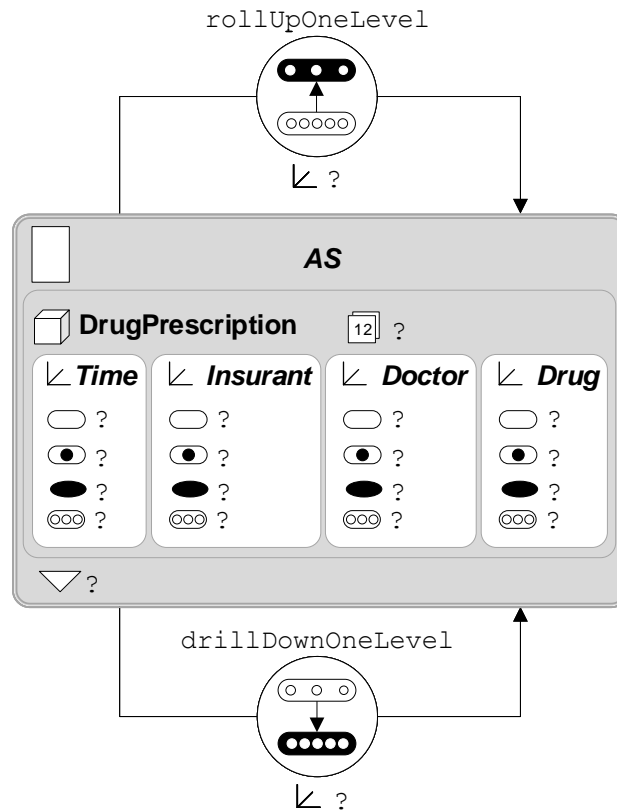


Figure 6.4: Example of a BI analysis graph schema comprising navigation step schemas having same analysis situation schema as source and target

any more. In this case of a BI analysis graph schema, other provinces are not of interest which could be a specific business requirement for the considered analysis task.

Figure 6.4 presents navigation step schemas having same analysis situation schema as source and target. There are two navigation step schemas, both with analysis situation schema AS as source and target, i.e., $|AS \Rightarrow AS| = 2$. One navigation step schema comprises operator `drillDownOneLevel`, the other one uses operator `rollUpOneLevel`. Hence, this is another example where the use of multi-graphs in the definition of BI analysis graph schemas is required.

Analysis situation schema AS consists almost exclusively of variables except for the cube and, of course, for dimension schemas used in dimension

qualifications. This non-comparative analysis situation schema can be considered as a pattern that allows to generate all queries on cube `DrugPrescription`. One can also think of a non-comparative analysis situation schema that specifies a simple query tool on cube `DrugPrescription`. The navigation step schemas allow to drill down and roll up one dimension level in an arbitrary dimension of cube `DrugPrescription`.⁵ Thus, one can also think of a non-comparative navigation step schema that specifies a simple OLAP tool on cube `DrugPrescription`.

In Figure 6.5, an example of a BI analysis graph schema is depicted having two navigation step schemas where the target analysis situation schema of one navigation step schema becomes the source of the other navigation step schema, and vice versa. This example comprises two edges between two vertices having opposite directions. Hence, we have the following sets and cardinalities: $|AS-1 \Rightarrow AS-2| = 1$ and $|AS-2 \Rightarrow AS-1| = 1$. In the first case, navigation operator `moveToNode` changes the dice node at level *year* of an analysis situation generated by *AS-1* with respect to dimension schema *Time* to another year in analysis situation instantiated from *AS-2*. The second navigation step schema generates navigation steps comprising operator `refocusSliceCond` that changes the slice condition of analysis situation *AS-2*, concerning dimension schema *Insurant*, to another slice condition (given by a variable) in analysis situation *AS-1*.

This small BI analysis graph schema of Figure 6.5 can be considered as a pattern that creates analysis situations and navigation steps by starting with an instance of analysis situation schema *AS-1* where variables are bound to a specific year and maybe to a specific set of slice conditions (unequal to empty set) with respect to dimension schema *Insurant*. The query result is listed per insurants' province for a certain year and, afterwards, one moves to another year (analysis situation schema *AS-2*). A business analyst would like to consider another restriction concerning dimension schema *Insurant* and

⁵We use variables for the single parameter of navigation operators `drillDownOneLevel` and `rollUpOneLevel`. Note, variables for dimension schema parameters are allowed in our definition of navigation step schemas. Without these dimension schema variables, one would have to define a separate navigation step schema for each dimension schema and navigation operator.

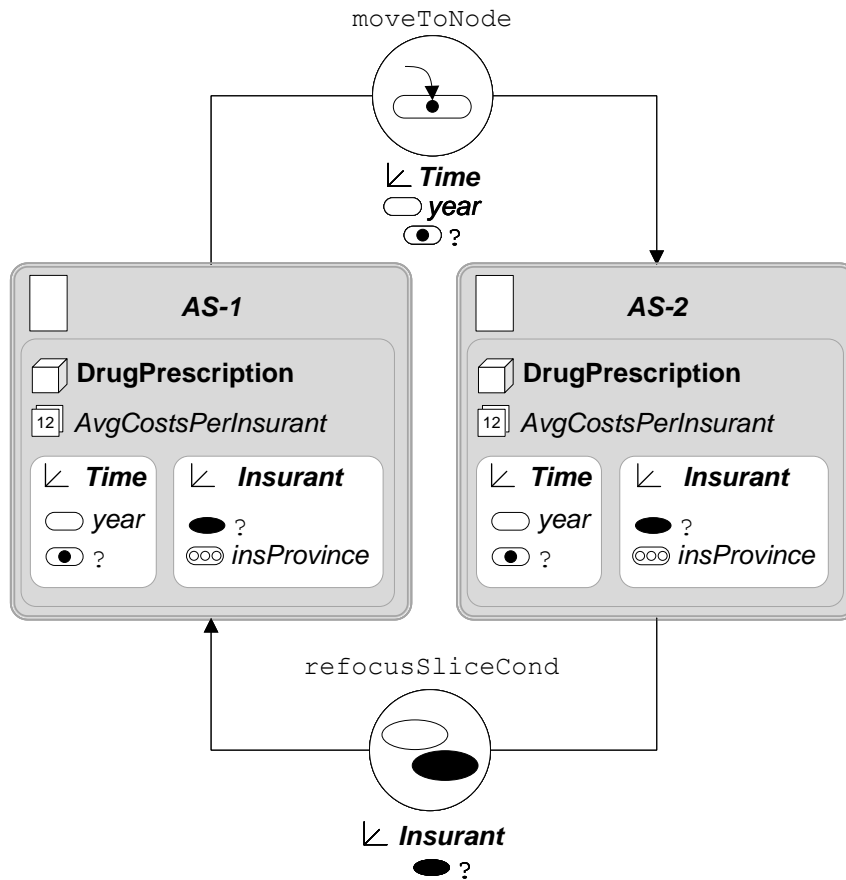


Figure 6.5: Example of a BI analysis graph schema having two navigation step schemas where the target analysis situation schema of one navigation step schema becomes the source of the other navigation step schema, and vice versa.

navigates back to *AS-1* by refocusing the set of slice conditions. Later we will see that similar behavior can also be achieved implicitly by backtracking.

Further examples of BI analysis graph schemas can be found in the previous Chapter 5. Although, some examples are only depicted for demonstration of navigation step schemas and do not yield a meaningful or complete application, they all can be considered as BI analysis graph schemas. Even single analysis situation schemas as presented in Figures 5.1–5.8 represent BI analysis graph schemas accordingly to Definition 6.7. All these examples contain

one vertex and no edges. Figure 5.9 represents a BI analysis graph schema only containing non-comparative navigation step schemas. A BI analysis graph schema that only consists of one comparative navigation step schema is depicted in Figure 5.12. Similar to the example of Figure 6.4, in Figure 5.13, a BI analysis graph schema is depicted that comprises navigation step schemas having same analysis situation schema as source and target. Simple examples of BI analysis graph schemas containing navigation guards are demonstrated in Figure 5.14 and Figure 5.15. All navigation patterns presented in Section 5.3.3 of Chapter 5 can be considered as further examples of specific BI analysis graph schemas to solve specific analysis tasks by general approaches.

After introducing the basic concepts of BI analysis graphs and BI analysis graph schemas, in the subsequent sections of this chapter, we continue with further concepts. The instantiation process describes how a BI analysis graph can be considered as an instance of a BI analysis graph schema. An analysis process yields an analysis trace where also backtracking is allowed. Moreover, advanced concepts are presented: subgraphs of BI analysis graphs and BI analysis graph schemas, and composite analysis situations.

6.3 Instances of Analysis Graph Schemas

This section gives a definition of an instance of a BI analysis graph schema. Each named analysis situation and each navigation step must be mapped to a named analysis situation schema and a navigation step schema in a compatible way. We will see that such mappings are graph homomorphisms.

Definition 6.8. A BI analysis graph \mathbf{ag} is an *instance of a BI analysis graph schema* AG , if

1. for each $\mathbf{nas} \in ASituations(\mathbf{ag})$, there exists a $NAS \in ASSchemas(AG)$ such that $as_{\mathbf{nas}}$ is an instance of AS_{NAS} and
2. for each $\mathbf{nav} \in NavSteps(\mathbf{ag})$, there exists a $NAV \in NavSchemas(AG)$ such that $NavStep_{\mathbf{nav}}$ is an instance of $NavStepSchema_{NAV}$. \square

The definition claims a mapping of named analysis situations of BI analysis graph \mathbf{ag} to named analysis situation schemas of BI analysis graph schema AG such that the analysis situation is an instance of the analysis situation schema. On the other side, also each navigation step of \mathbf{ag} has to be mapped to a navigation step scheme of AG such that the navigation step is an instance of the navigation step schema.

Note that in the inverse direction, the definition requires neither that each analysis situation schema must be mapped to an analysis situation nor that each navigation step schema must be mapped to a navigation step. This means that it is not forced that all analysis situation schemas and navigation step schemas must be used in the instantiation process for generating BI analysis graphs from the underlying BI analysis graph schema.

The mapping of a navigation step \mathbf{nav} to a navigation step schema NAV such that $NavStep_{\mathbf{nav}}$ is an instance of $NavStepSchema_{NAV}$ implies that also the source $Source_{\mathbf{nav}}$ and the target $Target_{\mathbf{nav}}$ have to be mapped to $Source_{NAV}$ and $Target_{NAV}$ such that $SourceAS_{\mathbf{nav}}$ is an instance of $SourceASSchema_{NAV}$ and $TargetAS_{\mathbf{nav}}$ is an instance of $TargetASSchema_{NAV}$. The following theorem states the existence of a graph homomorphism from BI analysis graph instance \mathbf{ag} to its scheme AG .

Theorem 6.1. If BI analysis graph \mathbf{ag} is an instance of a BI analysis graph schema AG , then there exists a graph homomorphism from \mathbf{ag} to AG . \square

Proof: We have to show that there exists a mapping H from $ASituations(\mathbf{ag})$ to $ASSchemas(AG)$ such that, if $e \in NavSteps(\mathbf{ag})$, then there exists an $E \in NavSchemas(AG)$ such that $H(Source_e) = Source_E$ and $H(Target_e) = Target_E$.

The first condition of Definition 6.8 assures that each vertex $\mathbf{nas} \in ASituations(\mathbf{ag})$ can be mapped to a vertex of $ASSchemas(AG)$. Let H be such a function from $ASituations(\mathbf{ag})$ to $ASSchemas(AG)$.

The second condition of Definition 6.8 assures that each edge $\mathbf{nav} \in NavSteps(\mathbf{ag})$ can be mapped to an edge $NAV \in NavSchemas(AG)$ such that $NavStep_{\mathbf{nav}}$ is an instance of $NavStepSchema_{NAV}$. Thus, H can be defined in such a way that $H(Source_{\mathbf{nav}}) = Source_{NAV}$ and $H(Target_{\mathbf{nav}}) = Target_{NAV}$.

which yields a graph homomorphism from \mathbf{ag} to AG . \square

The subsequent theorem assures that if we have consecutive navigation steps in a BI analysis graph instance, then one also obtains consecutive navigation step schemas in the associated BI analysis graph schema.

Theorem 6.2. Let BI analysis graph \mathbf{ag} be an *instance of a BI analysis graph schema* AG . If $\mathbf{nav}_1, \mathbf{nav}_2 \in \text{NavSteps}(\mathbf{ag})$ with $\text{Target}_{\mathbf{nav}_1} = \text{Source}_{\mathbf{nav}_2}$, then there exists $NAV_1, NAV_2 \in \text{NavSchemas}(AG)$ such that \mathbf{nav}_1 and \mathbf{nav}_2 are instances of NAV_1 and NAV_2 , and $\text{Target}_{NAV_1} = \text{Source}_{NAV_2}$. \square

Proof: The definition of an instance of a BI analysis graph schema assures that there exists $NAV_1, NAV_2 \in \text{NavSchemas}(AG)$ such that \mathbf{nav}_1 and \mathbf{nav}_2 are instances of NAV_1 and NAV_2 . We have to show that $\text{Target}_{NAV_1} = \text{Source}_{NAV_2}$. Let H be a graph homomorphism from \mathbf{ag} to AG that has to exist accordingly to Theorem 6.1. Then we can conclude $\text{Target}_{NAV_1} = H(\text{Target}_{\mathbf{nav}_1}) = H(\text{Source}_{\mathbf{nav}_2}) = \text{Source}_{NAV_2}$. \square

Our definition claims that each analysis situation and each navigation step is mapped to an analysis situation schema and to a navigation step schema by a compatible way that induces a homomorphism. Note, we do not have same requirements in the reverse direction. There might be analysis situation schemas that are not mapped to analysis situations of the BI analysis graph instance, i.e., an analysis situation schema need not be used when instantiating a BI analysis graph schema. On the other side, an analysis situation schema can be mapped to many analysis situations which will be often the case, i.e., analysis situation schemas are expected for instantiation of many analysis situations.

More formally, a homomorphism H from a BI analysis graph instance \mathbf{ag} to its BI analysis graph schema AG need neither be surjective nor injective. If $NAS \in \text{ASSchemas}(AG)$, then $H^{-1}(NAS)$ can contain zero, one, or more analysis situations. If $H^{-1}(NAS) = \emptyset$, surjectivity is violated, i.e., named analysis situation schema NAS is not used for instantiation. If $|H^{-1}(NAS)| > 1$, injectivity is violated, i.e., more than one analysis situations are generated from NAS .

Moreover, the definition of a BI analysis graph schema does not require

a start analysis situation schema. Each analysis situation schema can be chosen for the first instantiation to obtain the first analysis situation. In contrast, a BI analysis graph (except the empty graph) always has a start analysis situation which is the root of the directed tree.

BI analysis graph in Figure 6.1 (in the subsequent presentation denoted as *ag*) is an example of an instance of BI analysis graph schema of Figure 6.2 (denoted as *AG*). The graph homomorphism from *ag* to *AG* maps *as-1* to *AS-1*, *as-2* to *AS-2*, *as-3* to *AS-3*, *as-4.1* and *as-4.2* to *AS-4*, *as-5* to *AS-5*, *as-6* to *AS-6*, *as-7.1* and *as-7.2* to *AS-7*, and *as-8.1* and *as-8.2* to *AS-8*. There are analysis situation schemas that are not instantiated, for instance, *AS-9*. Analysis situations schemas *AS-4*, *AS-7*, and *AS-8* are instantiated more than once. One can easily check that all analysis situations are instances of the mapped analysis situation schema, for example, *as-1* is an instance of *AS-1* that is constructed by binding free variable *?y* to dimension node 2016.

To assure that *ag* is an instance of *AG*, one also has to check that each navigation step can be mapped to a navigation step schema such that the navigation step is an instance of the schema, for example, navigation step *as-1* \rightarrow *as-2* is an instance of the single navigation step schema in *AS-1* \Rightarrow *AS-2* and is mapped to it. There are two navigation steps *as-3* \rightarrow *as-4.1* and *as-3* \rightarrow *as-4.2* that are mapped to the single navigation step schema in *AS-3* \Rightarrow *AS-4*. They are instantiated by binding unbound parameter *?prov* to dimension nodes *Upper Austria* and *Vienna*. Navigation step *as-4.1* \rightarrow *as-5* is mapped to the navigation step schema in *AS-4* \Rightarrow *AS-5* that comprises a navigation guard. Because the navigation guard evaluates to true this navigation step is valid and can be invoked.

Further examples of BI analysis graph schemas and instances were also incorporated in the previous chapters. For instance, the navigation steps depicted in Figure 4.1 of Chapter 4 represent a BI analysis graph that is an instance of the example demonstrated in Figure 5.9 of Chapter 5 which represents a BI analysis graph schema. Navigation patterns explained in Chapter 5 can be also considered as small BI analysis graph schemas. For example, Figure 5.16 shows a BI analysis graph schema that allows to drill down along the level hierarchy of dimension schema *Insurant*. Moreover, the

example presented in Figure 5.17 represents a BI analysis graph that is an instance of the BI analysis graph schema of Figure 5.16.

More examples of instances of the BI analysis graph schema of Figure 6.2 can be constructed by binding the free variable to other values. For instance, if one binds variable `?y` in analysis situation schema *AS-1* to year 2018 and/or variable `?prov` in the single navigation step schema of set *AS-3* \Rightarrow *AS-4* to province Lower Austria instead of Upper Austria, one obtains other instances. Furthermore, Lower Austria could be analyzed additionally to Upper Austria and Vienna. In this case, an instance will be generated that contains additional branches generated by the navigation step schema of set *AS-3* \Rightarrow *AS-4*.

Figure 6.6 shows another instance of the BI analysis graph schema presented in Figure 6.2. This example demonstrates obviously that it is not necessary to use all analysis situation schemas and navigation step schemas of a BI analysis graph schema. Comparative analysis situation *as-4* is the root of the depicted BI analysis graph and represents an instance of analysis situation schema *AS-4*. In this example, instances of analysis situation schemas *AS-1*, *AS-2*, and *AS-3* are missing. In analysis situation schema *AS-4*, variables `?y` and `?prY` are bound to years 2018 and 2015, and variable `?prov` is bound to province Upper Austria. Note that we do not use the navigation step schema of set *AS-1* \Rightarrow *AS-2* including navigation operator `moveToPrevNode` with respect to dimension schema *Time*. Thus, in the instance of Figure 6.6, it is not forced that variable `?prY` contains the preceding year of year `?y`. In the example of Figure 6.6, variable `?prov` is only bound to Upper Austria and the subsequent instantiations only yield two branches of consecutive navigation steps (*as-4* \rightarrow *as-5* \rightarrow *as-6* and *as-4* \rightarrow *as-7* \rightarrow *as-8*).⁶

Although, it is not necessary to use all analysis situation schemas and navigation step schemas of a BI analysis graph schema, the analysis graph instance has to be a connected graph. For instance, let *ag* denote BI analysis graph of Figure 6.1 and let *AS* denote the BI analysis graph schema of

⁶Instead of listing navigation steps *as-4* \rightarrow *as-5* and *as-5* \rightarrow *as-6*, and *as-4* \rightarrow *as-7* and *as-7* \rightarrow *as-8*, we allow to use the more compact notation *as-4* \rightarrow *as-5* \rightarrow *as-6* and *as-4* \rightarrow *as-7* \rightarrow *as-8*, respectively.

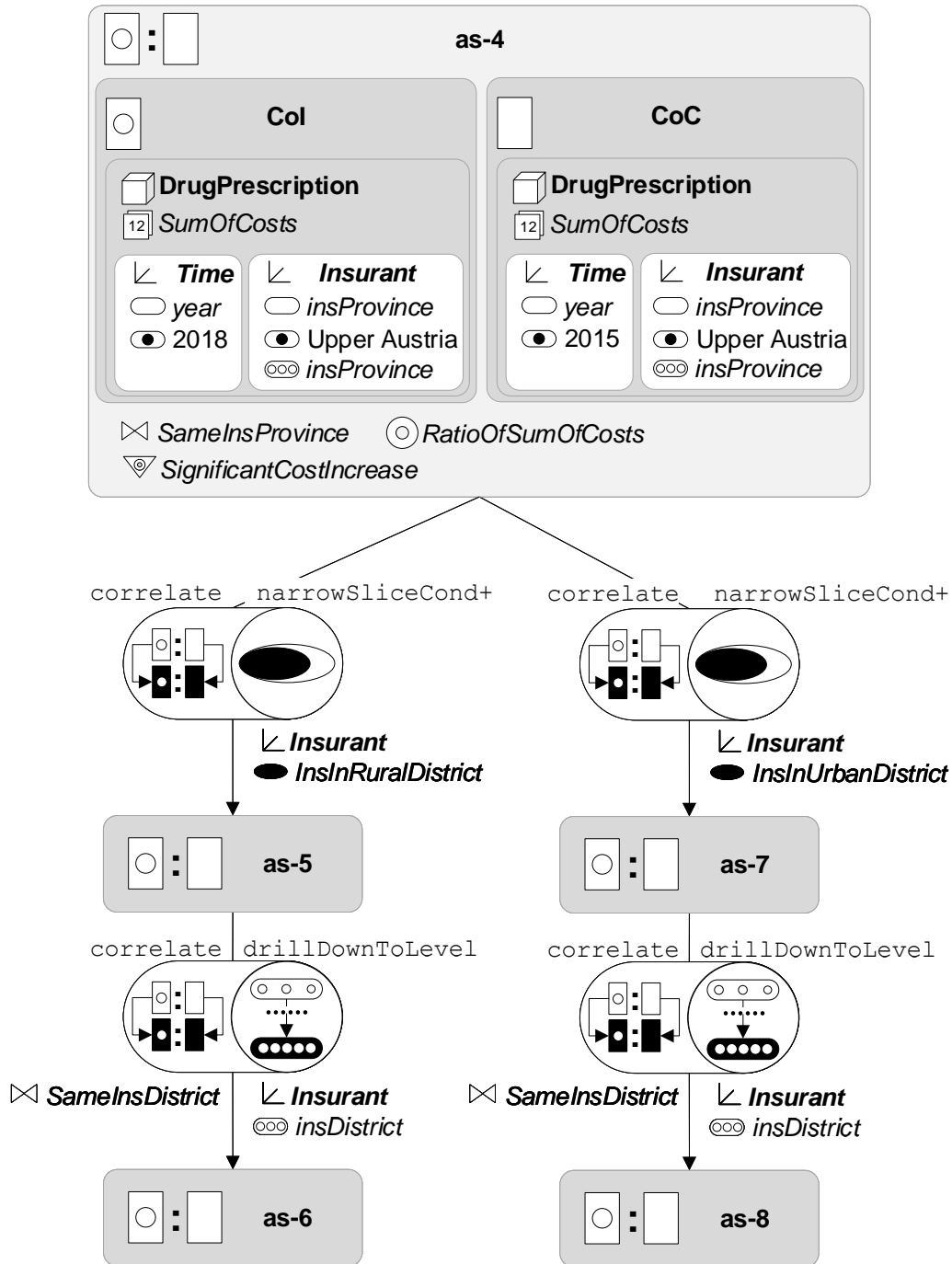


Figure 6.6: A smaller instance of BI analysis graph graph schema of Figure 6.1

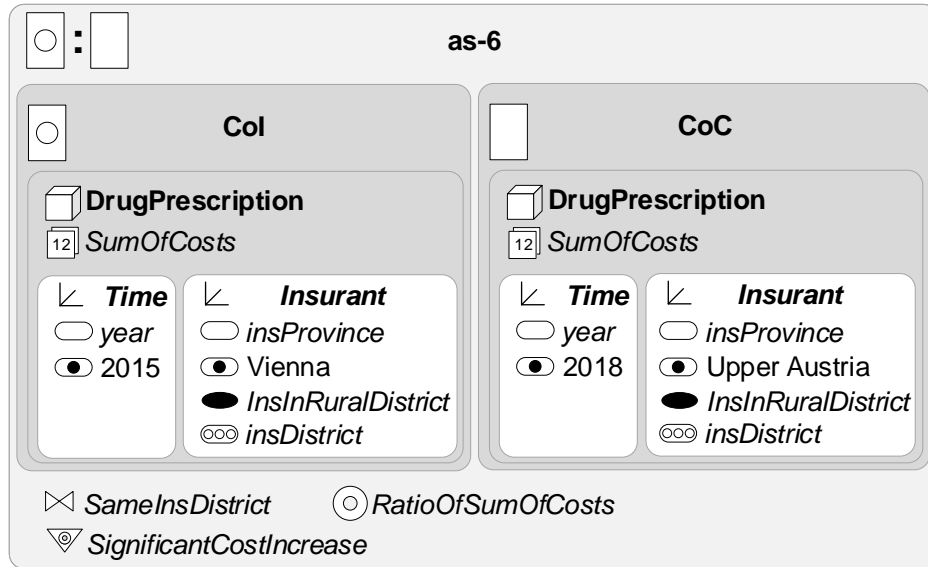


Figure 6.7: An instance of a BI analysis graph graph schema of Figure 6.1 consisting of only one vertex

Figure 6.2. Furthermore, we define BI analysis graphs ag_1 and ag_2 such that $ASituations_{ag_1} = \{as-1, as-2\} \subset ASituations_{ag}$, $NavSteps_{ag_1} = \{as-1 \rightarrow as-2\} \subset NavSteps_{ag}$, $ASituations_{ag_2} = \{as-3, as-4.1, as-4.2, as-5, as-6, as-7.1, as-7.2, as-8.1, as-8.2\} \subset ASituations_{ag}$, and $NavSteps_{ag_2} = \{as-3 \rightarrow as-4.1, as-3 \rightarrow as-4.2, as-4.1 \rightarrow as-5 \rightarrow as-6, as-4.1 \rightarrow as-7.1 \rightarrow as-8.1, as-4.2 \rightarrow as-7.2 \rightarrow as-8.2\} \subset NavSteps_{ag}$. BI analysis graph ag as well as BI analysis graphs ag_1 and ag_2 are instances of BI analysis graph schema AS . However, the unions of analysis situations and navigation steps of ag_1 and ag_2 , $ASituations_{ag_1} \cup ASituations_{ag_2}$ and $NavSteps_{ag_1} \cup NavSteps_{ag_2}$, do not yield a BI analysis graph because there is not a connection (navigation step) between $as-2$ and $as-3$, i.e., the resulting graph is not connected.

In Figure 6.7, a particular case of an instance of the BI analysis graph schema of Figure 6.2 is presented. It only contains one vertex and no edges. Analysis situation $as-6$ can be considered as an instance of analysis situation schema $AS-6$. Notice that there is also an unusual variable assignment with respect to the whole BI analysis graph schema. The year variable in the context of interest is bound to 2015 and the province to Vienna, whereas in

the context of comparison the year is bound to 2018 and the province to Upper Austria. Also remember, that only rural districts are compared and, moreover, a district in the context of interest is compared with the same district in the context of comparison. As one can convince, this example of an analysis situation is not meaningful. Actually, because of the assignments of the province variables, this analysis situation yields empty result sets.

The example in the previous paragraph also shows that navigation steps (and navigation step schemas) include additional semantics to an analysis process. The defined navigation step schemas of Figure 6.2 ensure that only a certain year is compared with its previous year, that districts of the same province are compared, and that, for province Vienna, restriction to rural districts is excluded because this restriction does not make sense. Navigation operators and navigation guards are used to define appropriate semantics for an analysis process. If the mapping from an instance to its BI analysis graph schema represents a surjective homomorphism (an epimorphism), all constituents a the BI analysis graph schema (analysis situation schemas and navigation step schemas) are also instantiated in the retrieved BI analysis graph. In this case, one can casually say that all of the semantics of the BI analysis graph schema is transferred to its instance or all of the specified semantics of the BI analysis graph schema is also used in the instantiated BI analysis graph that represents an analysis process.

Figure 6.8 demonstrates an instance of the BI analysis graph schema of Figure 6.4. Whereas in Figure 6.2 a BI analysis graph schema with specific semantics (specifying specific analysis processes) is represented, the BI analysis graph schema of Figure 6.4 specifies very general tasks (rolling up and drilling down with respect to arbitrary dimensions of drug prescriptions). One can think of a specification of a very simple OLAP query tool. In Figure 6.8, an exemplary application of such a tool is demonstrated. A user defines the first analysis situation *as-1*. She or he selects aggregate measure *SumOfCosts*, year 2018, and granularity levels *insProvince* and *docDistrict*, i.e., drug prescription costs are listed for year 2018 per insurants' province and doctors' district. Afterwards, the user drills down to insurants' districts (analysis situation *as-2*), rolls up to doctors' provinces (analysis situation

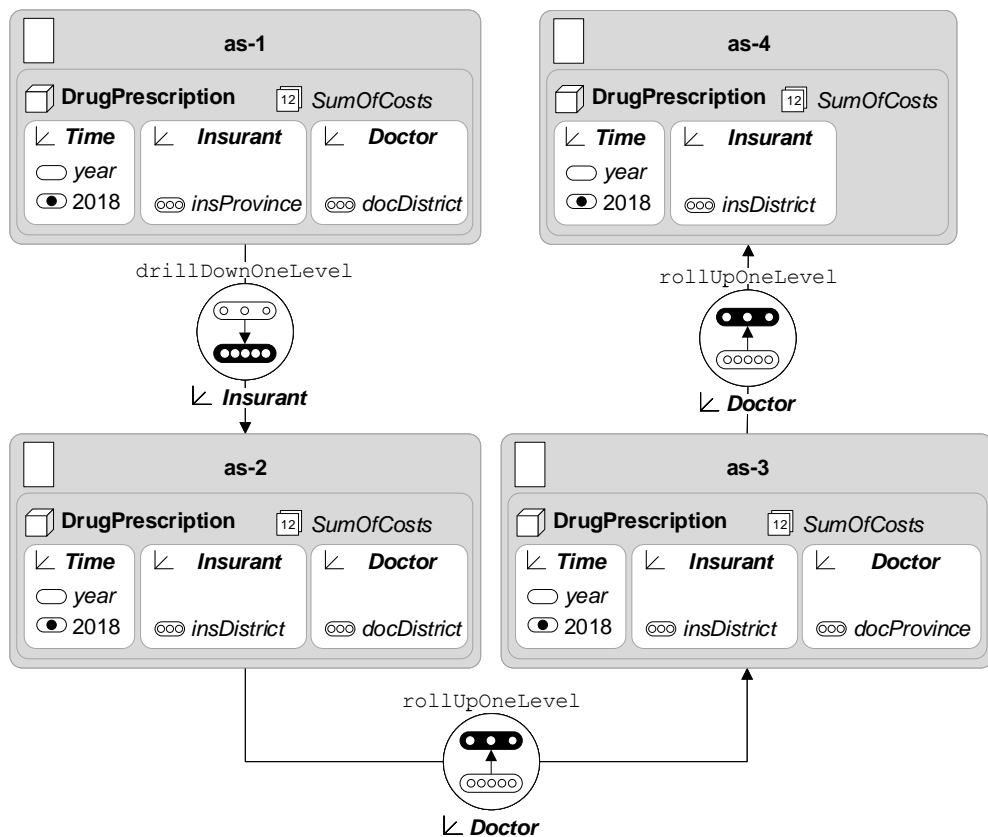


Figure 6.8: Instance of BI analysis graph schema of Figure 6.4

as-3), and, finally, she or he omits dimension schema *Doctor* by rolling up to dimension level *top* (analysis situation as-4).

In the subsequent sections of this chapter, advanced concepts of BI analysis graphs and BI analysis graph schemas are presented. The following two sections describes the possibility to structure BI analysis graphs and BI analysis graph schemas by subgraphs. Composite analysis situations represent a specific kind of subgraphs at instance and schema level comprising additional conditions. Finally, in another section, considerations about the temporal order concerning query execution induced by a BI analysis graph are presented.

6.4 Structuring by Subgraphs

To preserve an overview in comprehensive processes, one usually divides a process into smaller subprocesses in a hierarchical manner. This principle can be also applied to analysis processes. Because analysis processes are defined as graphs at schema level as well as at instance level (BI analysis graph schema and BI analysis graph), subprocesses for analysis (OLAP sessions) can be defined as subgraphs of BI analysis graph schemas or BI analysis graphs.

Subgraphs provide simple means of subject-oriented hierarchical structuring. To assure that such a subgraph is again a BI analysis graph schema or a BI analysis graph, we only allow connected subgraphs. Obviously, a subgraph of a BI analysis graph schema or a BI analysis graph is again a BI analysis graph schema or a BI analysis graph, respectively.

Definition 6.9. SG is a *subgraph of a BI analysis graph schema* AG (written as $SG \sqsubseteq AG$), if SG is a connected subgraph of AG . \square

Theorem 6.3. A subgraph SG of a BI analysis graph schema AG is a BI analysis graph schema. \square

Proof: AG is a connected directed multi-graph and SG is a connected graph with $ASSchemas(SG) \subseteq ASSchemas(AG)$ and $NavStepSchemas(SG) \subseteq NavStepSchemas(AG)$. Thus, one can show that also SG satisfies the properties of a BI analysis graph schema. \square

Definition 6.10. sg is a *subgraph of a BI analysis graph* ag (written as $sg \sqsubseteq ag$), if sg is a connected subgraph of ag . \square

Theorem 6.4. A subgraph sg of a BI analysis graph ag is a BI analysis graph. \square

Proof: As ag is a directed tree and sg is a connected graph with $ASituations(sg) \subseteq ASituations(ag)$ and $NavSteps(sg) \subseteq NavSteps(ag)$, one can show that also sg satisfies the properties of a BI analysis graph. \square

Graphically, a decomposition into subgraphs can be depicted as a tree. We use the same visualization for both decomposition of a BI analysis graph

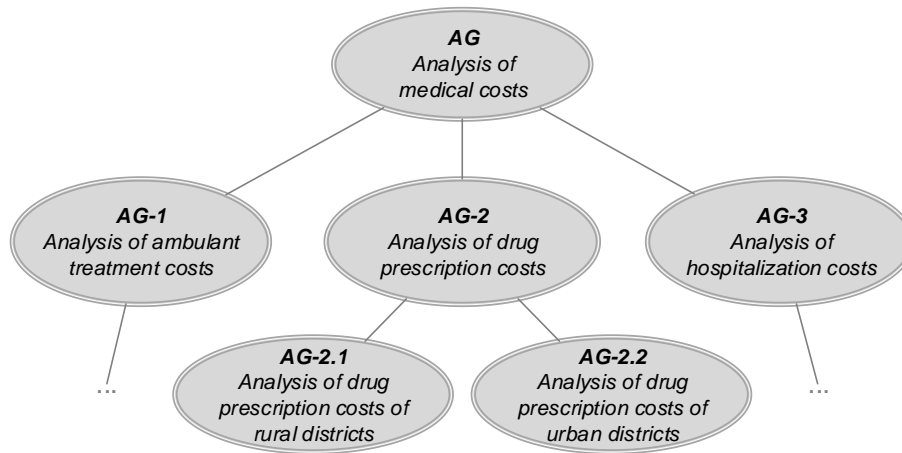


Figure 6.9: Hierarchical structuring of a BI analysis graph schema by subgraphs (corresponding to the example of Figure 6.2)

and decomposition of a BI analysis graph schema; apart from that, subgraph symbols of the tree of subgraphs of BI analysis graph schemas are drawn by double-edged borders. Figure 6.9 shows a possible decomposition of BI analysis graph schema in Figure 6.2. The whole BI analysis graph schema is denoted by *AG* (analysis of medical costs). *AG* is decomposed into subgraphs *AG-1* (analysis of ambulant treatment costs), *AG-2* (analysis of drug prescription costs), and *AG-3* (analysis of hospitalization costs).⁷ In Figure 6.10, the proposed decomposition of BI analysis graph schema of Figure 6.2 is visualized by delimited navigation step schemas using frames with different colors. This decomposition corresponds to the tree representation of subgraphs in Figure 6.9. Thus, the same labels can be found in both Figure 6.9 and Figure 6.10.⁸

As one can see in Figure 6.10, subgraph *AG-2* comprises analysis situation schemas *AS-1*, *AS-2*, *AS-3*, *AS-4*, *AS-5*, *AS-6*, *AS-7*, and *AS-8* and navigation step schemas $AS-1 \Rightarrow AS-2$, $AS-2 \Rightarrow AS-3$, $AS-3 \Rightarrow AS-4$, $AS-4 \Rightarrow AS-5$,

⁷We only focus on drug prescription costs (subgraph *AG-2*). Cost analysis for ambulant treatments (subgraph *AG-1*) and hospitalizations (subgraph *AG-3*) can be defined in a similar way.

⁸Note, subgraph *AG-1* and subgraph *AG-3* are not completely depicted in both Figure 6.9 and Figure 6.10. They are only roughly intimated in both figures.

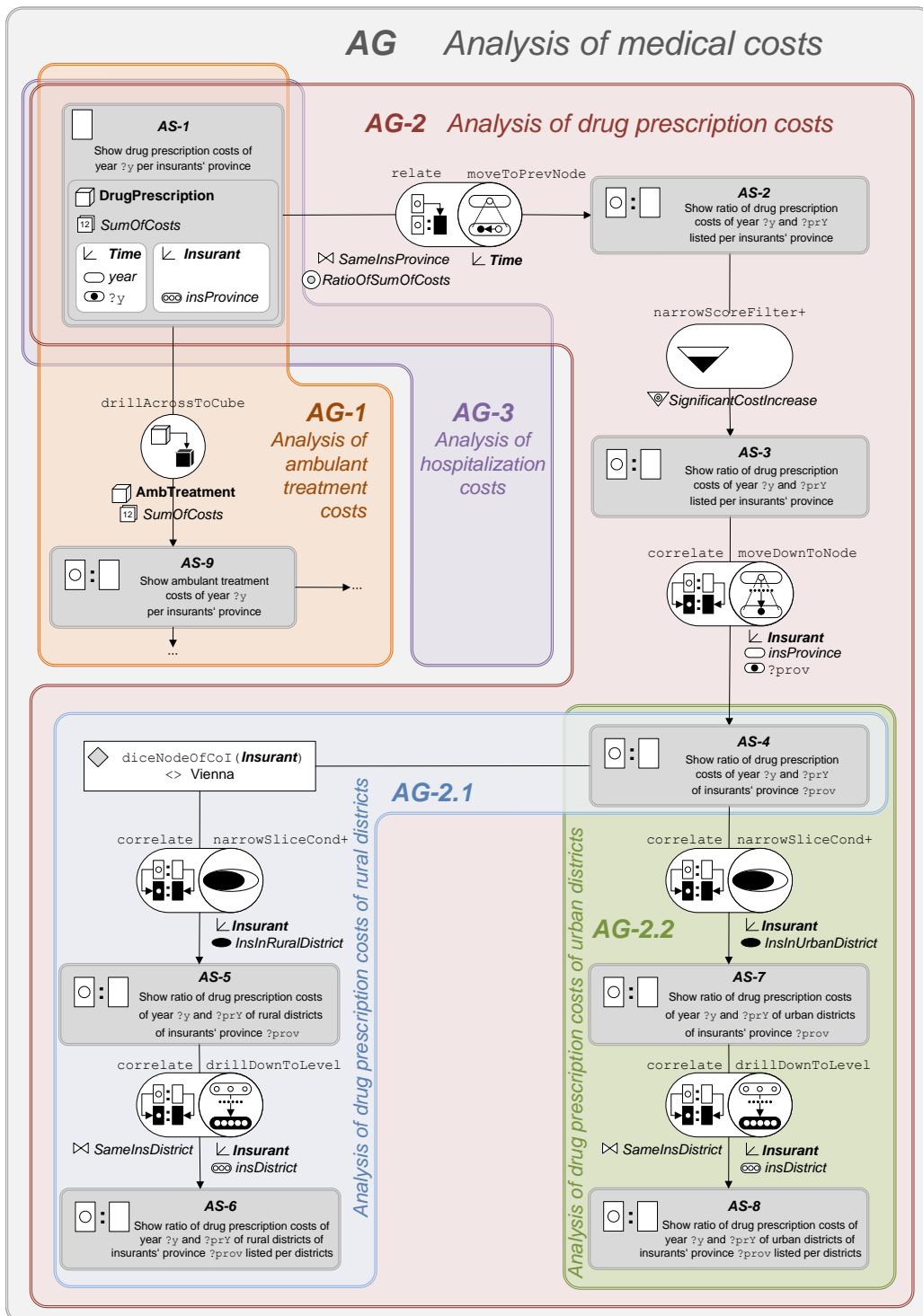


Figure 6.10: Visual decomposition of BI analysis graph schema of Figure 6.2 into subgraphs (corresponding to the tree representation in Figure 6.9)

$AS-5 \Rightarrow AS-6$, $AS-4 \Rightarrow AS-7$, and $AS-7 \Rightarrow AS-8$. Moreover, BI analysis graph schema $AG-2$ is decomposed in $AG-2.1$ (analysis of drug prescription costs of rural districts) and $AG-2.2$ (analysis of drug prescription costs of urban districts) such that $AG-2.1 \sqsubseteq AG-2$ and $AG-2.2 \sqsubseteq AG-2$. BI analysis graph schema $AG-2.1$ comprises analysis situation schemas $AS-4$, $AS-5$, and $AS-6$, and navigation step schemas $AS-4 \Rightarrow AS-5 \Rightarrow AS-6$, and BI analysis graph schema $AG-2.2$ comprises analysis situation schemas $AS-4$, $AS-7$, and $AS-8$, and navigation step schemas $AS-4 \Rightarrow AS-7 \Rightarrow AS-8$.

Note, a decomposition of a graph into subgraphs (or a decomposition of a subgraph into further subgraphs) need neither be complete nor disjoint. The first property means, that the union of all subgraphs need not yield the original graph. For example, the union of $AG-2.1$ and $AG-2.2$ does not contain $AS-1$, $AS-2$, $AS-3$, $AS-4$, $AS-1 \Rightarrow AS-2$, $AS-2 \Rightarrow AS-3$, and $AS-3 \Rightarrow AS-4$ but which are contained in graph $AG-2$. The second property means that the intersection of the vertices and edges of two subgraphs need not be empty, i.e., there may exist analysis situation vertices or edges that belong to both subgraphs. For example, BI analysis graph schemas $AG-2.1$ and $AG-2.2$ comprise the common analysis situation schema $AS-4$.

Analogously to the schema level, we demonstrate an example of subgraphs at instance level. Figure 6.11 shows a hierarchical structure of BI analysis graph (denoted as \mathbf{ag}) of Figure 6.1 which is an instance of BI analysis graph schema in Figure 6.2. BI analysis graphs $\mathbf{ag-1}$ (analysis of ambulant treatment costs of year 2016), $\mathbf{ag-2}$ (analysis of drug prescription costs of year 2016), and $\mathbf{ag-3}$ (analysis of hospitalization costs of year 2016) are subgraphs of BI analysis graph \mathbf{ag} (analysis of medical costs of year 2016).⁹ Moreover, subgraph $\mathbf{ag-2}$ is divided into subgraphs $\mathbf{ag-2.1}$ (analysis of drug prescription costs of rural districts of 2016 in Upper Austria), $\mathbf{ag-2.2.1}$ (analysis of drug prescription costs of urban districts of 2016 in Upper Austria), and $\mathbf{ag-2.2.2}$ (analysis of drug prescription costs of urban districts of 2016 in Vienna). Similar to the subgraph representation at schema level in Figure 6.10, Figure

⁹Again we focus on drug prescription costs (subgraph $\mathbf{ag-2}$). Cost analysis for ambulant treatments (subgraph $\mathbf{ag-1}$) and hospitalizations (subgraph $\mathbf{ag-3}$) can be defined in a similar way.

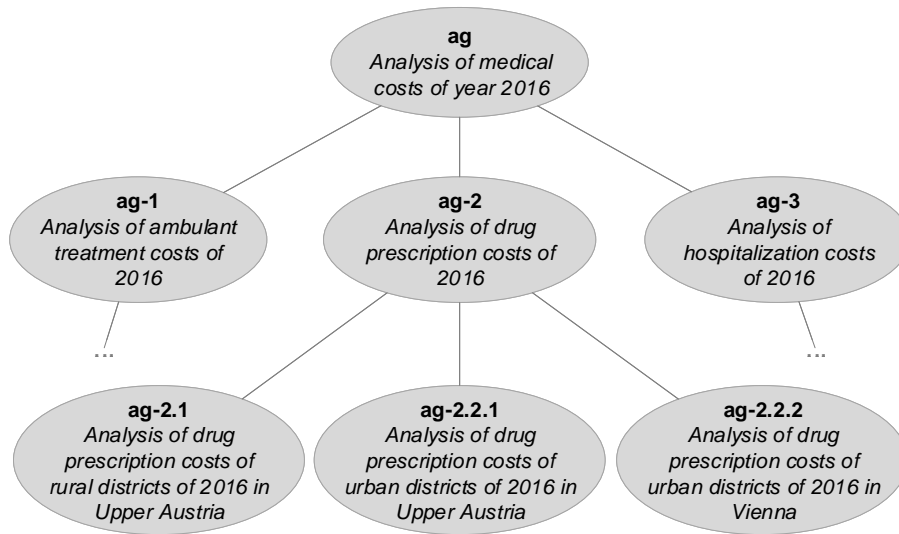


Figure 6.11: Hierarchical structuring of a BI analysis graph by subgraphs (corresponding to the example of Figure 6.1)

6.12 shows the decomposition of the BI analysis graph of Figure 6.1 which corresponds to the tree representation as presented in Figure 6.11. Again, in Figure 6.12 the various subgraphs are emphasized by colored frames and appropriate labels as used in Figure 6.11.¹⁰

As depicted in Figure 6.12, BI analysis graph *ag-2* comprises analysis situations *as-1*, *as-2*, *as-3*, *as-4.1*, *as-4.2*, *as-5*, *as-6*, *as-7.1*, *as-8.1*, *as-7.2*, and *as-8.2*, and navigation steps $as-1 \rightarrow as-2 \rightarrow as-3$, $as-3 \rightarrow as-4.1 \rightarrow as-5 \rightarrow as-6$, $as-3 \rightarrow as-4.1 \rightarrow as-7.1 \rightarrow as-8.1$, and $as-3 \rightarrow as-4.2 \rightarrow as-7.2 \rightarrow as-8.2$. Subgraph *ag-2.1* contains analysis situations *as-4.1*, *as-5*, and *as-6*, and navigation steps $as-4.1 \rightarrow as-5 \rightarrow as-6$, subgraph *ag-2.2.1* includes analysis situations *as-4.1*, *as-7.1*, and *as-8.1*, and navigation steps $as-4.1 \rightarrow as-7.1 \rightarrow as-8.1$, and subgraph *ag-2.2.2* comprises analysis situations *as-4.2*, *as-7.2*, and *as-8.2*, and navigation steps $as-4.2 \rightarrow as-7.2 \rightarrow as-8.2$. Analogously to schema level, one can see that a decomposition of a graph into subgraphs at instance level need neither be complete nor disjoint.

The following theorem reveals that a hierarchical structuring by sub-

¹⁰Note, subgraph *ag-1* and subgraph *ag-3* are completely omitted in the representation of Figure 6.12 to keep the depiction as simple as possible.

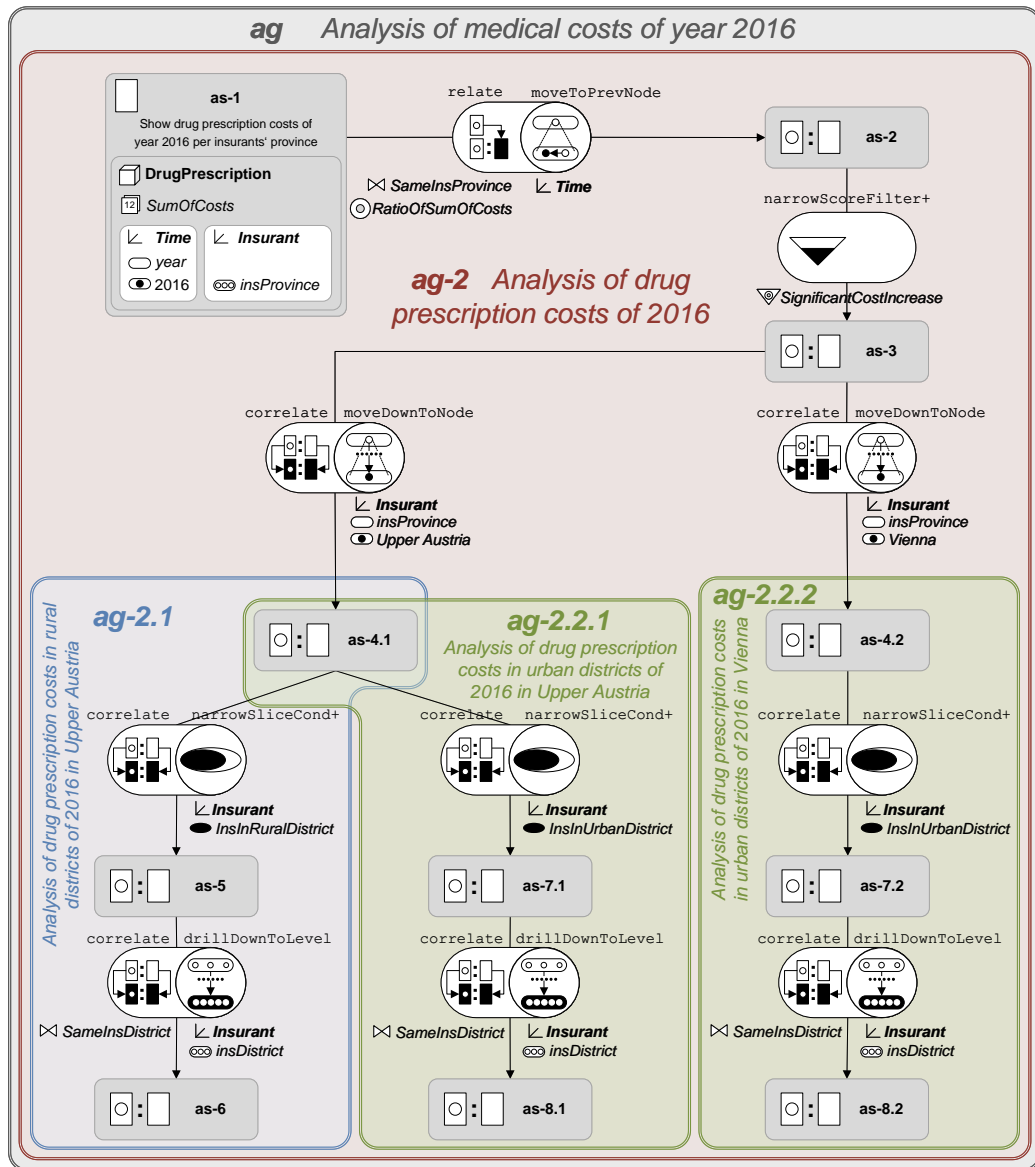


Figure 6.12: Visual decomposition of BI analysis graph of Figure 6.1 into subgraphs (corresponding to the tree representation in Figure 6.11)

graphs at schema level induces a hierarchical structuring at instance level. That means, a decomposition hierarchy at instance level can be generated from a decomposition hierarchy at schema level.

Theorem 6.5. If BI analysis graph \mathbf{ag} is an instance of BI analysis graph schema AG , $SG \sqsubseteq AG$, H is a graph homomorphism from \mathbf{ag} to AG , and \mathbf{sg} is an instance of SG with $ASituations(\mathbf{sg}) \subseteq ASituations(\mathbf{ag})$ and $NavSteps(\mathbf{sg}) \subseteq NavSteps(\mathbf{ag})$, then $\mathbf{sg} \sqsubseteq \mathbf{ag}$ and restriction $H|_{ASituations(\mathbf{sg})}$ is a graph homomorphism from \mathbf{sg} to SG . \square

Proof: From $ASituations(\mathbf{sg}) \subseteq ASituations(\mathbf{ag})$ and $NavSteps(\mathbf{sg}) \subseteq NavSteps(\mathbf{ag})$, and because \mathbf{sg} is a connected graph (\mathbf{sg} is a BI analysis graph which must be connected), proposition $\mathbf{sg} \sqsubseteq \mathbf{ag}$ follows directly. Moreover, one can show that $H|_{ASituations(\mathbf{sg})}$ remains a graph homomorphism. \square

For further explanations, we consider the hierarchical structuring of a BI analysis graph schema and a BI analysis graph as depicted in Figure 6.9 and Figure 6.11, and as previously specified in this section. Note that this example refers to BI analysis graph schema presented in Figure 6.2 and to BI analysis graph shown in Figure 6.1. The following example focuses on BI analysis graph schemas $AG-2$ and $AG-2.1$, and on BI analysis graphs $\mathbf{ag-2}$ and $\mathbf{ag-2.1}$. In this example, $\mathbf{ag-2}$ is an instance of BI analysis graph schema $AG-2$, $\mathbf{ag-2.1}$ is an instance of BI analysis graph schema $AG-2.1$, $AG-2.1 \sqsubseteq AG-2$, $ASituations(\mathbf{ag-2.1}) \subseteq ASituations(\mathbf{ag-2})$, and $NavSteps(\mathbf{ag-2.1}) \subseteq NavSteps(\mathbf{ag-2})$. Furthermore, we consider the graph homomorphism from $\mathbf{ag-2}$ to $AG-2$ as specified in Section 6.3, i.e., $\mathbf{as-1}$ is mapped to $AS-1$, $\mathbf{as-2}$ to $AS-2$, $\mathbf{as-3}$ to $AS-3$, $\mathbf{as-4.1}$ and $\mathbf{as-4.2}$ to $AS-4$, $\mathbf{as-5}$ to $AS-5$, $\mathbf{as-6}$ to $AS-6$, $\mathbf{as-7.1}$ and $\mathbf{as-7.2}$ to $AS-7$, and $\mathbf{as-8.1}$ and $\mathbf{as-8.2}$ to $AS-8$. Accordingly to Theorem 6.5, BI analysis graph $\mathbf{ag-2.1}$ has to be a subgraph of BI analysis graph $\mathbf{ag-2}$ ($\mathbf{ag-2.1} \sqsubseteq \mathbf{ag-2}$) and the graph homomorphism from $\mathbf{ag-2}$ to $AG-2$ can be restricted to $ASituations(\mathbf{ag-2.1})$ ($= \{\mathbf{as-4.1}, \mathbf{as-5}, \mathbf{as-6}\}$) yielding a graph homomorphism from $\mathbf{ag-2.1}$ to $AG-2.1$, i.e., analysis situation $\mathbf{as-4.1}$ is mapped to analysis situation schema $AS-4$, $\mathbf{as-5}$ is mapped to $AS-5$, and $\mathbf{as-6}$ to $AS-6$.

In Figure 6.9 and 6.11, relationships between subgraphs of BI analysis

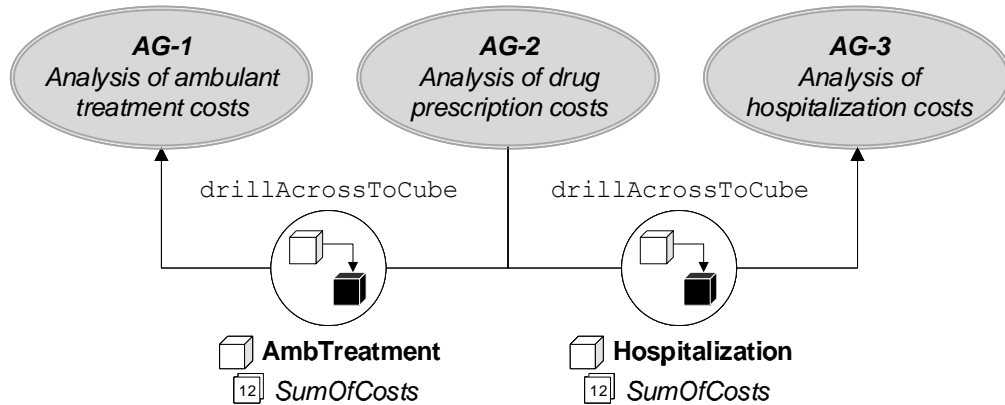


Figure 6.13: Embedded subgraphs *AG-1*, *AG-2*, and *AG-3* at schema level

graph schemas and BI analysis graphs are depicted in a hierarchical order. Specific symbols are used to draw subgraphs in a condensed graphical representation: double edged ellipses for BI analysis graph schemas and single edged ellipses for BI analysis graphs. Figures 6.13–6.19 show how the condensed graphical representation of subgraphs can be graphically embedded into the encompassed graph.

Figure 6.13 shows a BI analysis graph schema for analyzing medical costs. Subgraph *AS-2* comprises analysis of drug prescription costs. Analysis of ambulant treatment costs and hospitalization costs are defined by BI analysis graph schemas *AG-1* and *AG-3*. Both subgraphs are connected to subgraph *AG-2* by navigation step schemas including navigation operator *drillAcrossToCube*.¹¹ In case of ambulant treatment costs, the navigation step schema contains source analysis situation schema *AS-1* that belong to subgraph *AG-2* and target analysis situation schema *AS-9* belonging to subgraph *AG-1*. The navigation step schema from *AS-1* to *AS-9* neither belong to subgraph *AG-1* nor to subgraph *AG-2* but it combines both subgraphs. Similar considerations can be made with respect to the connection between subgraph *AG-2* and subgraph *AG-3* (analysis of hospitalization costs).

¹¹In this example of Figure 6.13, we make a slight difference to decomposition as depicted in Figure 6.10. In Figure 6.10, analysis situation schema *AS-1* and the outgoing navigation step schemas to subgraphs *AG-1* and *AG-3* are elements of *AG-1* and *AG-3*, respectively, whereas in Figure 6.13, they do not represent elements of *AG-1* and *AG-3*.

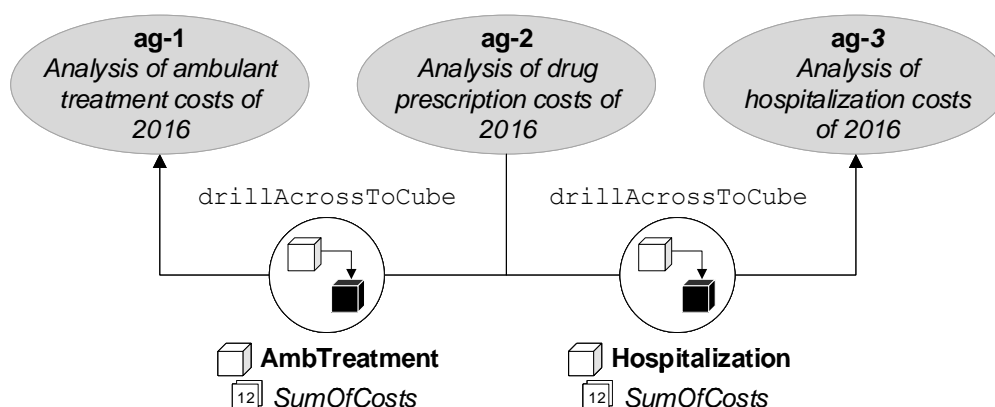


Figure 6.14: Embedded subgraphs **ag-1**, **ag-2**, and **ag-3** at instance level

Figure 6.14 depicts the analogous situation at instance level. BI analysis graphs **as-1**, **as-2**, and **as-3** represent instances of BI analysis graph schemas **AS-1**, **AS-2**, and **AS-3**, respectively. Costs of drug prescriptions, ambulant treatments, and hospitalizations are analyzed for year 2016. The depicted connections comprising navigation operator **drillAcrossToCube** represent navigation step instances of the navigation step schemas presented in Figure 6.13.

In Figure 6.15, another example of embedded subgraphs at schema level is presented. One can see subgraphs **AG-1** and **AG-3** that are connected by navigation step schemas with source analysis situation schema **AS-1** and with navigation operator **drillAcrossToCube**. It represents a similar context as already shown in Figure 6.13. But there are two more subgraphs: BI analysis graph schema **AG-2.1** for analyzing drug prescription costs of rural districts and BI analysis graph schema **AG-2.2** for analyzing drug prescription costs of urban districts. Both BI analysis graph schemas include analysis situation schema **AS-4**. In addition to it, BI analysis graph schema **AG-2.1** contains analysis situation schemas **AS-5** and **AS-6**, and navigation step schemas $AS-4 \Rightarrow AS-5 \Rightarrow AS-6$, and BI analysis graph schema **AG-2.2** additionally comprises analysis situation schemas **AS-7** and **AS-8**, and navigation step schemas $AS-4 \Rightarrow AS-7 \Rightarrow AS-8$. Because analysis situation schema **AS-4** is a member of both subgraph **AG-2.1** and subgraph **AG-2.2**, a

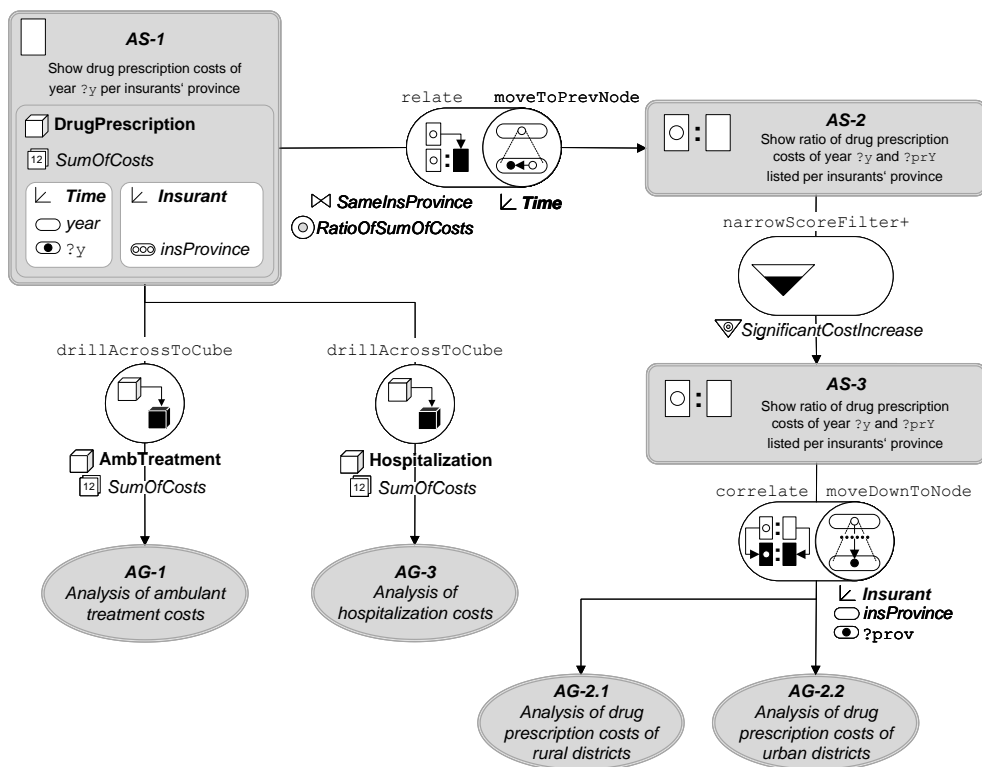


Figure 6.15: Embedded subgraphs AG-2.1 and AG-2.2 at schema level

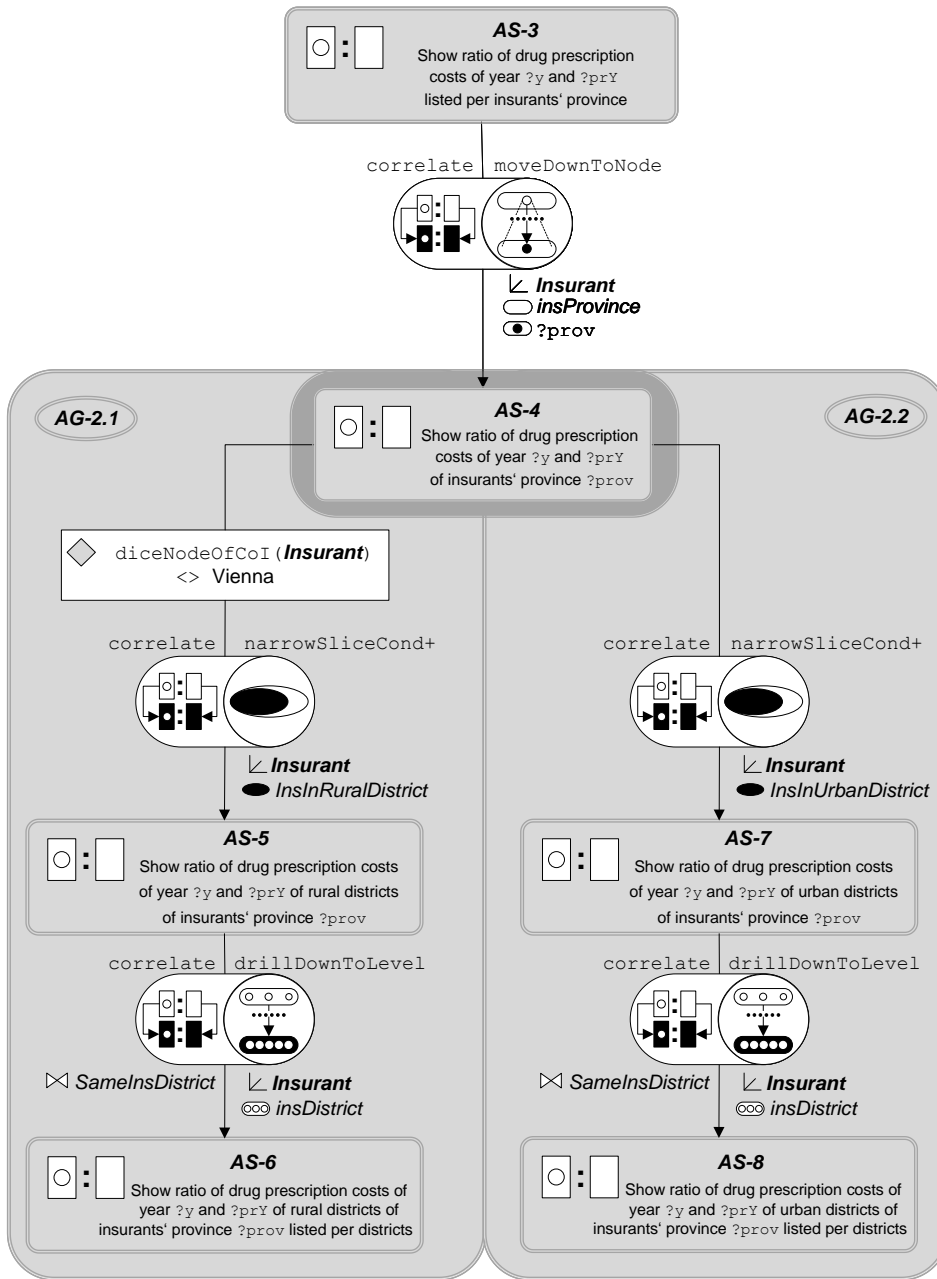


Figure 6.16: Boundary between subgraphs AG-2.1 and AG-2.2

fork is drawn from the operator symbol to both subgraphs. But note, there is only one navigation step schema from analysis situation schema $AS-3$ to analysis situation schema $AS-4$. Figure 6.16 presents an inside view of both subgraphs. One can see that analysis situation schema $AS-4$ is depicted only once. The dark greyed background around analysis situation schema $AS-4$ symbolizes that this analysis situation schema belongs to both subgraphs. In this drawing, the graphical fork can be avoided.

Figure 6.17 shows a variant of separating subgraphs of BI analysis graph schema of Figure 6.2. Subgraphs $AG-B.2.1$ and $AG-B.2.2$ do not contain a common analysis situation schema (analysis situation schema $AS-4$) as presented in Figures 6.15 and 6.16. BI analysis graph schema $AG-B.2.1$ comprises analysis situation schemas $AS-5$ and $AS-6$, and navigation step schema $AS-5 \Rightarrow AS-6$, and BI analysis graph schema $AG-B.2.2$ comprises analysis situation schemas $AS-7$ and $AS-8$, and navigation step schema $AS-7 \Rightarrow AS-8$. It depends on business analyst's point of view how to decompose a BI analysis graph schema into subgraphs.

Examples of instances of the BI analysis graph schemas of Figure 6.15 and Figure 6.17 including subgraphs are presented in Figure 6.18 and Figure 6.19. In Figure 6.18, subgraphs $ag-2.1$, $ag-2.2.1$, and $ag-2.2.2$ are depicted. It is assumed that, accordingly to Figure 6.1, subgraph $ag-2.1$ comprises analysis situations $as-4.1$, $as-5$, $as-6$, and navigation steps $as-4.1 \rightarrow as-5 \rightarrow as-6$ which are instances of analysis situation schemas $AS-4$, $AS-5$, $AS-6$, and navigation step schemas $AS-4 \Rightarrow AS-5 \Rightarrow AS-6$; subgraph $ag-2.2.1$ comprises analysis situations $as-4.1$, $as-7.1$, $as-8.1$, and navigation steps $as-4.1 \rightarrow as-7.1 \rightarrow as-8.1$ which are instances of analysis situation schemas $AS-4$, $AS-7$, $AS-8$, and navigation step schemas $AS-4 \Rightarrow AS-7 \Rightarrow AS-8$; and subgraph $ag-2.2.2$ comprises analysis situations $as-4.2$, $as-7.2$, $as-8.2$, and navigation steps $as-4.2 \rightarrow as-7.2 \rightarrow as-8.2$ which are also instances of analysis situation schemas $AS-4$, $AS-7$, $AS-8$, and navigation step schemas $AS-4 \Rightarrow AS-7 \Rightarrow AS-8$. Subgraph $ag-2.1$ is an instance of BI analysis graph schema $AG-2.1$ whereas subgraphs $ag-2.2.1$ and $ag-2.2.2$ are instances of BI analysis graph schema $AG-2.2$. Note, BI analysis graphs $ag-2.1$ and $ag-2.2.1$ have $as-4.1$ in common.

In Figure 6.19, instances of BI analysis graph schemas $AG-B.2.1$ and

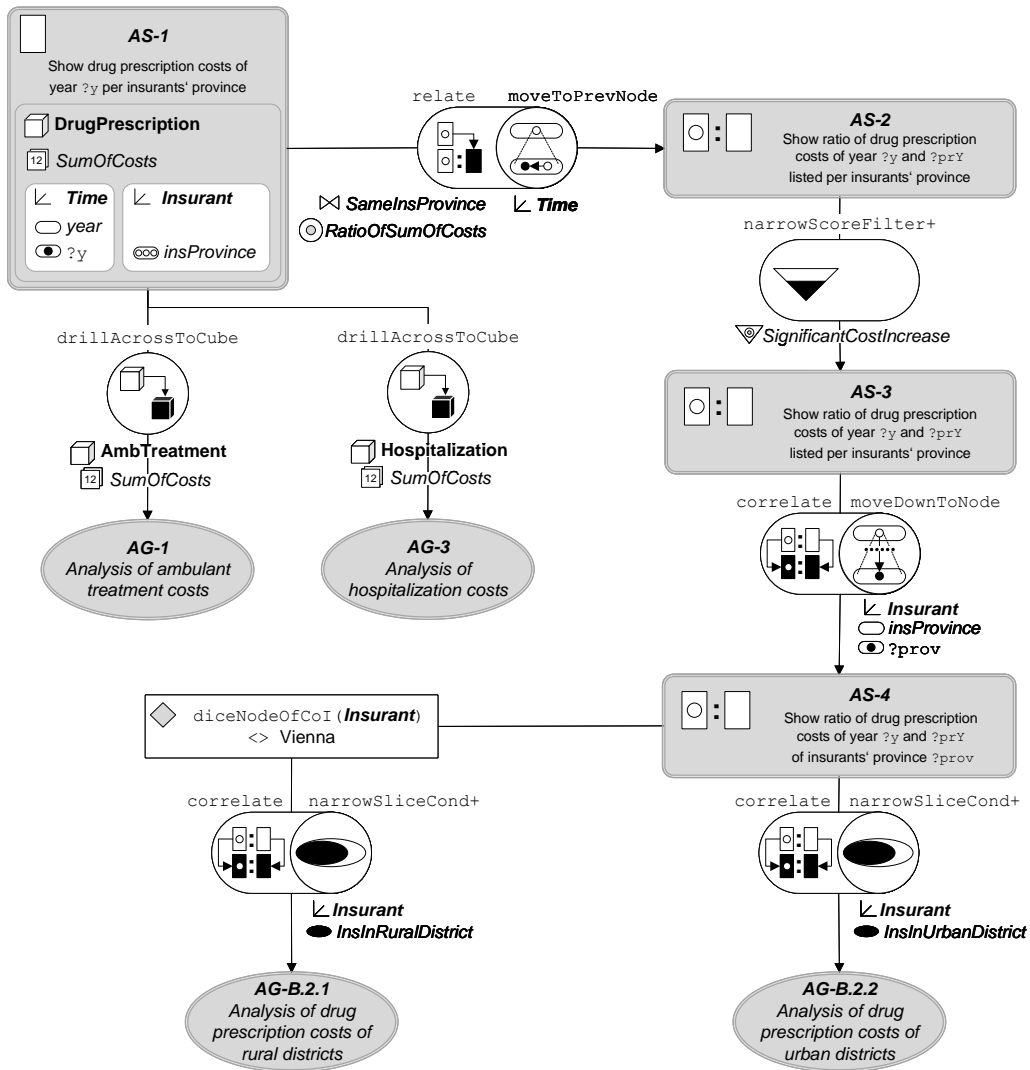


Figure 6.17: Variants AG-B.2.1 and AG-B.2.2 of embedded subgraphs at schema level

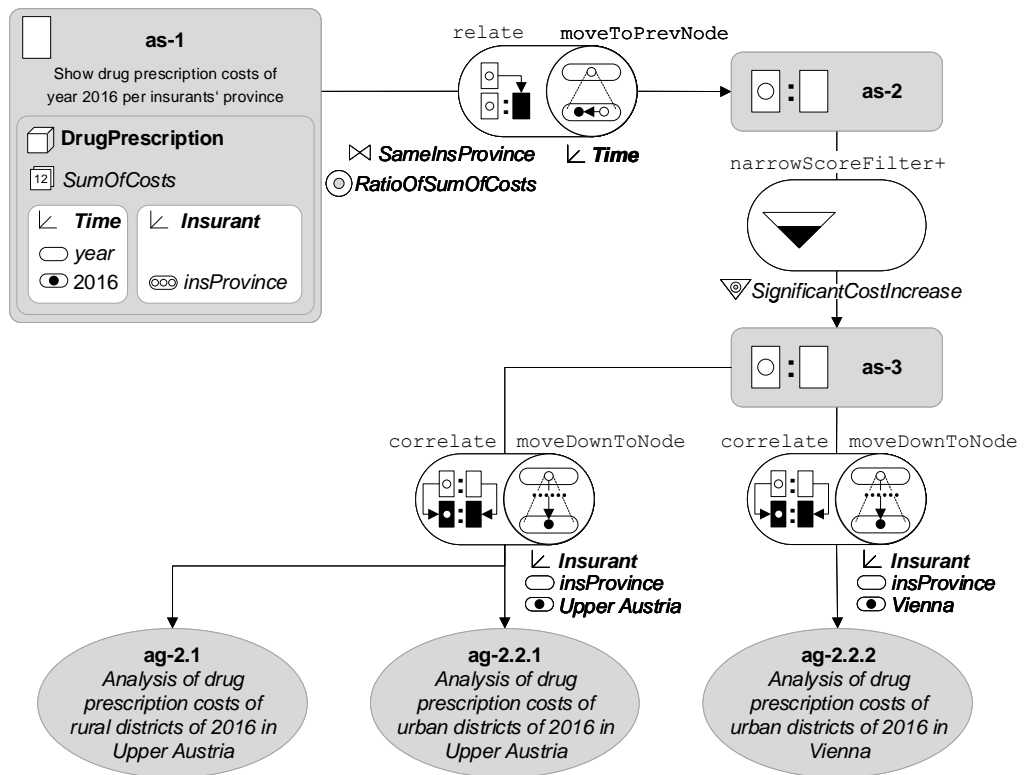


Figure 6.18: Embedded subgraphs ag-2.1, ag-2.2.1, and ag-2.2.2 at instance level

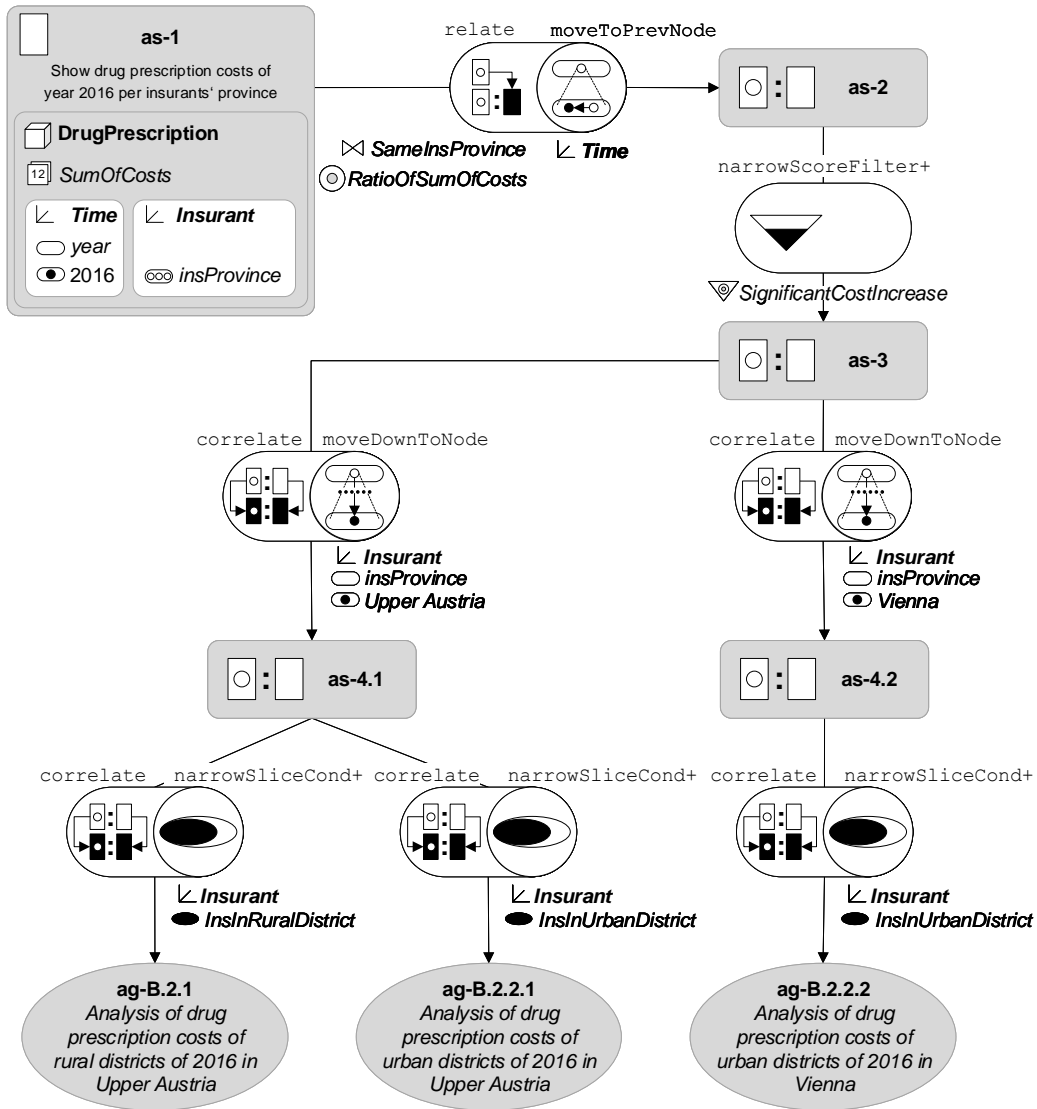


Figure 6.19: Embedded subgraphs ag-B.2.1, ag-B.2.2.1, and ag-B.2.2.2 at instance level

AG-B.2.2 of Figure 6.17 are depicted. We assume that subgraph *ag-B.2.1* contains analysis situations *as-5*, *as-6*, and navigation step *as-5* \rightarrow *as-6*. It represents an instance of BI analysis graph schema *AG-B.2.1*. BI analysis graph *ag-B.2.2.1* is an instance of BI analysis graph schema *AG-B.2.2*. It comprises analysis situations *as-7.1* and *as-8.1*, and navigation step *as-7.1* \rightarrow *as-8.1*. Analysis situations *as-7.2* and *as-8.2*, and navigation step *as-7.2* \rightarrow *as-8.2* are included in subgraph *ag-B.2.2.2* that represents another instance of BI analysis graph schema *AG-B.2.2*. In this example, subgraphs *ag-B.2.1*, *ag-B.2.2.1*, and *ag-B.2.2.2* do not contain a common analysis situation.

In the next section, composite analysis situations are introduced at schema and instance level. They represent a specific kind of subgraphs (including additional restrictions) of BI analysis graph schemas and BI analysis graphs. The principle of hierarchical structuring by subgraphs can also be applied to composite analysis situation schemas and composite analysis situations.

6.5 Composite Analysis Situation

Up to now, we have discussed analysis situations as single multi-dimensional non-comparative and comparative queries. A subgraph of a BI analysis graph comprises analysis situations that can be considered as a collection of queries which represents a composite analysis situation. Such a subgraph can also be represented at schema level as a composite analysis situation schema.

As a difference to arbitrary subgraphs of a BI analysis graph schema, we claim that the instantiation of a composite analysis situation schema induces the unique and finite instantiation of all contained analysis situation schemas in one go, i.e., a navigation step schema to a composite analysis situation schema must assure that all free variables of the target composite analysis situation schema are bound. In a conceptual view, an analysis situation schema cannot be instantiated partially. Thus, also a composite analysis situation schema can only be instantiated as a whole. As a consequence, within a composite analysis situation schema, no new unbound variables can be introduced by a navigation step schema. With respect to WebML, such navigation step schemas within a composite analysis situation schema can

be compared with automatic links that transfer information automatically without further user input.

To assure that a composite analysis situation schema can be instantiated as a whole, we require a root analysis situation schema as a part of the composite. All incoming navigation step schemas must have the root analysis situation schema as a target. And from the root, all other analysis situation schemas of the composite must be reachable. Moreover, we have to ensure that, starting from the root, the instantiation of a composite analysis situation schema can be uniquely and finitely determined. Thus, it is necessary that all free variables can be uniquely bound and that instantiations of a infinite number of analysis situations caused by specific graph cycles are prevented.

In the subsequent subsection, preliminary definitions are introduced to facilitate the definition of composite analysis situation schemas. The second and third subsections formally define composite analysis situation schemas (at schema level) and composite analysis situations (at instance level). In the last subsection, the graphical representation and examples of composite analysis situation schemas and composite analysis situations are presented.

6.5.1 Preliminary Definitions

Before defining composite analysis situation schemas, we introduce preliminary definitions. Composite analysis situation schemas are subgraphs that are embedded in a BI analysis graph schema. One can consider the part of such a BI analysis graph schema that do not belong to an embedded composite analysis situation schema as the environment of this composite analysis situation schema.

The subsequent two definitions specify the environment of a general subgraph of a BI analysis graph schema. The set of environmental analysis situation schemas and the set of environmental navigation step schemas contain those analysis situation schemas and navigation step schemas that do not belong the subgraph. To be precise, in the case of environmental navigation step schemas, it is required that both the source analysis situation schema

and the target analysis situation schema must not belong to the subgraph, or, in other words, both the source analysis situation schema and the target analysis situation schema must belong to the set of environmental analysis situation schemas.

Definition 6.11. The set of environmental analysis situation schemas $EnvASSchemas_{AG}(SG)$ of subgraph SG of BI analysis graph schema AG is defined as $EnvASSchemas_{AG}(SG) = ASSchemas_{AG} - ASSchemas_{SG}$. \square

Definition 6.12. The set of environmental navigation step schemas $EnvNavSchemas_{AG}(SG)$ of subgraph SG of BI analysis graph schema AG is defined as $EnvNavSchemas_{AG}(SG) = \{NAV \in NavSchemas_{AG} \mid Source_{NAV} \in EnvASSchemas_{AG}(SG) \text{ and } Target_{NAV} \in EnvASSchemas_{AG}(SG)\}$. \square

Note, although set $EnvNavSchemas_{AG}(SG)$ contains navigation step schemas comprising analysis situation schemas of $EnvASSchemas_{AG}(SG)$ as source and target, both sets $EnvNavSchemas_{AG}(SG)$ and $EnvASSchemas_{AG}(SG)$ do not necessarily define a connected subgraph of AG . Therefore, both sets together do not necessarily define another BI analysis graph schema.

Composite analysis situation schemas are subgraphs that should be considered on the same level as non-comparative or comparative analysis situation schemas. There are navigation step schemas that lead to a composite analysis situation schema having the composite analysis situation schema as target and there are navigation step schemas going out from a composite analysis situation schema having the composite analysis situation schema as source. Such navigation step schemas represent a boundary crossing between a composite analysis situation schema and its environment.

The following four definitions specify the boundary (as a kind of interface) between a general subgraph of a BI analysis graph schema and its environment. Input navigation step schemas and output navigation step schemas represent an input and output interface, respectively. Whereas input navigation step schemas comprise a source analysis situation schema that belong to the environment and a target analysis situation schema that belong to the subgraph, source analysis situation schemas of output navigation step

schemas are members of the subgraph and target analysis situation schemas of output navigation step schemas belong to the set of environmental analysis situation schemas. The targets of input navigation step schemas represent the set of input analysis situation schemas and the sources of output navigation step schemas represent the set of output analysis situation schemas.

Definition 6.13. The *set of input navigation step schemas* $InNavSchemas_{AG}(SG)$ of subgraph SG of analysis graph schema AG is defined as $InNavSchemas_{AG}(SG) = \{NAV \in NavSchemas_{AG} \mid Source_{NAV} \in EnvNavSchemas_{AG}(SG) \text{ and } Target_{NAV} \in ASSchemas_{SG}\}$. \square

Definition 6.14. The *set of input analysis situation schemas* $InASSchemas_{AG}(SG)$ of subgraph SG of analysis graph schema AG is defined as $InASSchemas_{AG}(SG) = \{Target_{NAV} \mid NAV \in InNavSchemas_{AG}(SG)\}$. \square

Definition 6.15. The *set of output navigation step schemas* $OutNavSchemas_{AG}(SG)$ of subgraph SG of analysis graph schema AG is defined as $OutNavSchemas_{AG}(SG) = \{NAV \in NavSchemas_{AG} \mid Source_{NAV} \in ASSchemas_{SG} \text{ and } Target_{NAV} \in EnvNavSchemas_{AG}(SG)\}$. \square

Definition 6.16. The *set of output analysis situation schemas* $OutASSchemas_{AG}(SG)$ of subgraph SG of analysis graph schema AG is defined as $OutASSchemas_{AG}(SG) = \{Source_{NAV} \mid NAV \in OutNavSchemas_{AG}(SG)\}$. \square

Because variables provide a certain degree of freedom in the specification of BI analysis graph schemas (and subgraphs) in the sense that there is more than one unique instantiation caused by different variable bindings, we have to restrict the use of free variables in composite analysis situation schemas. For composite analysis situation schemas, free variables have to be bound in a unique way such that a composite analysis situation schema can be instantiated in one go. First we provide the following three preliminary definitions that specify the notion of free variables of non-comparative and comparative analysis situation schemas, and of navigation step schemas.

Definition 6.17. A non-comparative analysis situation schema AS has free variables, if (accordingly to Definition 5.1) $CubeInstance_{AS} = ?$, $BMsrsConds_{AS} = ?$, $AMsrs_{AS} = ?$, $FilterConds_{AS} = ?$, or, for $D \in DimSchemas_{AS}$, $DiceLvl_{AS}(D) = ?$, $DiceNode_{AS}(D) = ?$, $SliceConds_{AS}(D) = ?$, or $GranLvl_{AS}(D) = ?$, respectively. \square

For each component of a non-comparative analysis situation schema, a variable can be used that represents a free variable. The following definition introduces free variables of comparative analysis situations. Free variables of comparative analysis situation schemas can be free variables of the context of interest and the context of comparison that represent non-comparative analysis situation schemas. Additionally, variables used for the set of join conditions, the set of scores, or the set of score filters are also considered as free variables.

Definition 6.18. A comparative analysis situation schema CAS has free variables, if (accordingly to Definition 5.3) CoI_{CAS} or CoC_{CAS} has free variables, or $JoinConds_{CAS} = ?$, $Scores_{CAS} = ?$, or $ScoreFilters_{CAS} = ?$, respectively. \square

Free variables of navigation step schemas are free variables of the source analysis situation schema or variables used as actual parameters. This is expressed by the following definition.

Definition 6.19. A navigation step schema NAV defined (accordingly to Definition 5.6) as

- (a) $NAV = (SRC, NavGrd, OP(p_1, \dots, p_q), (\bar{p}_1, \dots, \bar{p}_q), TRG)$ or
- (b) $NAV = (SRC, NavGrd, OP(OP'(p'_1, \dots, p'_{q'}), p_1, \dots, p_q), ((\bar{p}'_1, \dots, \bar{p}'_{q'}), \bar{p}_1, \dots, \bar{p}_q), TRG)$

has free variables, if

1. SRC has free variables or
2. the list of actual parameters $(\bar{p}_1, \dots, \bar{p}_q)$ or $((\bar{p}'_1, \dots, \bar{p}'_{q'}), \bar{p}_1, \dots, \bar{p}_q)$, respectively, contains variables.

In the second case, we say: Navigation step schema NAV has free variables in the list of actual parameters. Free variables of source analysis situation schema SRC (in case 1) and free variables of the actual parameter list (in case 2) are also called free variables of navigation step schema NAV . \square

Note that in case of $Source_{NAV} \neq Target_{NAV}$ for navigation step schema NAV , free variables of the target analysis situation schema $Target_{NAV}$ do not represent free variables of a navigation step schema NAV . Free variables of the target analysis situation schema are already bound in the navigation step schema either by constants or variables of the source analysis situation schema accordingly to the frame assumption, or by the behavior of the navigation operator which is defined by the operator's pre- and post-condition (see Chapter 4), by the operator's actual parameters that can also contain free variables, and by the source analysis situation schema.

In the specific case of graph loops, we have $Source_{NAV} = Target_{NAV}$. Note, in this case, free variables of source $Source_{NAV}$ are also free variables of navigation step schema NAV (accordingly to the definition).¹²

6.5.2 Formal Definition at Schema Level

After the preliminary definitions of the previous subsection, in this subsection, composite analysis situation schemas are defined. Composite analysis situation schemas are subgraphs of a BI analysis graph schemas including additional restrictions and composite analysis situations are instances of such subgraphs comprising further restrictions.

Definition 6.20. A composite analysis situation schema $AS = (SG, R)$ of a BI analysis graph schema AG comprises

1. a subgraph SG of AG ($SG \sqsubseteq AG$) and
2. an analysis situation schema R such that

¹²But be aware that there may be another navigation step schema NAV' that has analysis situation schema $Target_{NAV}$ ($= Target_{NAV'}$) as target but not as source. In this case, free variables of analysis situation schema $Target_{NAV'}$ ($= Target_{NAV}$) are not free variables of navigation step schema NAV' .

- (a) $InASSchemas_{AG}(SG) \subseteq \{R\}$ and,
 - (b) for each $V \in ASSchemas_{SG}$, there exists a sequence of navigation step schemas (E_1, \dots, E_n) with $n \geq 1$ and, for $1 \leq i \leq n$, $E_i \in NavSchemas_{SG}$, and $Target_{E_i} = Source_{E_{i+1}}$, $Source_{E_1} = R$, and $Target_{E_n} = V$, and
3. there exists no navigation step schema $NAV \in NavSchemas_{SG}$ such that NAV has free variables in the list of actual parameters.

Analysis situation schema R is also called *root analysis situation schema* (or simply *root*) of the composite analysis situation schema AS and is denoted by $Root_{AS}$. Moreover, in the context of AS , we define $AGSchema_{AS} = SG$. \square

Root analysis situation schema $Root_{AS}$ ($= R$) of composite analysis situation schema AS of a BI analysis graph schema AG represents a dedicated vertex of subgraph $AGSchema_{AS}$ where the instantiation process starts. Restriction (2a) specifies that input navigation step schemas lead to the root analysis situation schema as target. There are no other input navigation step schemas that have other targets. Set $InASSchemas_{AG}(SG)$ is a subset of set $\{R\}$. This also allows that $InASSchemas_{AG}(SG)$ is empty which means that there are no input navigation step schemas to composite analysis situation schema AS at all. Because the root is part of the definition of AS , we have again a unique starting point for instantiation.

Note that in the case of $InASSchemas_{AG}(SG) \neq \emptyset$, the root is a target of one or more navigation step schemas outside of subgraph SG and, therefore, it does not comprise free variables. On the other side, in the case of $InASSchemas_{AG}(SG) = \emptyset$, the root analysis situation schema may comprise free variables that have to be bound at instantiation time.

Restriction (2b) of Definition 6.20 ensures that every vertex (analysis situation schema) V can be reached by a sequence of edges (navigation step schemas) starting at the root analysis situation schema. This restriction assures that the whole composite analysis situation schema can be instantiated in one go.

To ensure a deterministic instantiation process, we have to avoid to introduce free variables within the composite analysis situation schema except in

the case of $InASSchemas_{AG}(SG) = \emptyset$. If there are no input navigation step schemas, the root is allowed to have free variables that have to be bound at instantiation time.¹³ The only way to introduce free variables (except in the case of $InASSchemas_{AG}(SG) = \emptyset$) within the composite analysis situation schema would be to define them as actual parameters of navigation step schemas that belong to the composite analysis situation schema. Hence, restriction (3) of Definition 6.20 prevents to introduce free variables via the list of actual parameters of navigation step schemas.

6.5.3 Formal Definition at Instance Level

Composite analysis situations can be considered as instances of composite analysis situation schemas. The definition in this subsection expresses that not every instance of a composite analysis situation schema represents a composite analysis situation. Additional restrictions are required which ensure that every analysis situation schema can be instantiated in a unique way such that the whole composite analysis situation schema is instantiated in one go.

Definition 6.21. For composite analysis situation schema AS , an instance \mathbf{as} of $AGSchema_{AS}$ is called *composite analysis situation*, if the following conditions are satisfied:

1. The root of \mathbf{as} (also denoted as $Root_{\mathbf{as}}$) represents an instance of $Root_{AS}$.
2. There exists a surjective graph homomorphism H from \mathbf{as} to $AGSchema_{AS}$ and for all $NAV \in NavSchemas_{AS}$, there also exists a $\mathbf{nav} \in NavSteps_{\mathbf{as}}$ such that \mathbf{nav} is an instance of NAV .
3. For all $NAS \in ASSchemas_{AS}$ and for all $\mathbf{nas}_1, \mathbf{nas}_2 \in ASituations_{\mathbf{as}}$ that are instances of NAS , restriction $as_{\mathbf{nas}_1} \neq as_{\mathbf{nas}_2}$ is satisfied.

¹³One can think of that a user has two options to instantiate a composite analysis situation schema. In the first case, she or he navigates via an input navigation step schema to the root of the composite analysis situation schema, and, in the second case, if input navigation step schemas are missing (or not used), the user has to bind all free variables of the root. All other analysis situation schemas and navigation step schemas of a composite analysis situation schema have to be instantiated automatically as a whole and in one go.

4. For all $\mathbf{nav} \in NavSteps_{as}$ with \mathbf{nav} is an instance of navigation step schema $NAV \in NavSchemas_{AS}$ and for all navigation step \mathbf{nav}' with $Source_{\mathbf{nav}'} = Target_{\mathbf{nav}}$ which is an instance of a navigation step schema NAV' such that $NAV' \in NavSchemas_{AS}$ and $Source_{NAV'} = Target_{NAV}$, it holds that also $\mathbf{nav}' \in NavSteps_{as}$ except condition 3 is violated. \square

The first condition of Definition 6.21 ensures that the root of the composite analysis situation represents an instance of the root of the composite analysis situation schema. Note that a composite analysis situation is a BI analysis graph (subgraph) which is defined as a directed tree. Thus, the root of a composite analysis situation corresponds with the notion of the root of a tree.

Definition 6.8 also allows that the instantiation of a BI analysis graph need not use all analysis situation schemas and navigation step schemas of the corresponding BI analysis graph schema. In contrast, for composite analysis situations, it is required that all analysis situation schemas and navigation step schemas of the composite analysis situation schema are used for instantiation.¹⁴ Thus, the second condition of Definition 6.21 says that an instance of a composite analysis situation schema represents a composite analysis situation, if there exists a surjective graph homomorphism and if all navigation step schemas are also instantiated.¹⁵ This condition expresses that, for every analysis situation schema of the composite analysis situation

¹⁴As discussed in Chapter 5, note that each navigation step schema and each instance of a navigation step schema always comprise a navigation guard (see Definition 5.6 and Definition 5.7). If a navigation guard for an instance of a navigation step schema evaluates to false, one also could say that “there is no instance at all” or “the instantiation process is aborted”, or the like. But note, these are informal considerations used for simplified presentations. Nevertheless, accordingly to Definition 5.7, an instance of a navigation step schema always includes a navigation guard and, especially in cases where properties of the source analysis situation are examined, there are also navigation guards that always evaluate to false at instance level. Thus, the claim that all analysis situation schemas and navigation step schemas of the composite analysis situation schema are used for instantiation can be justified (also in the case that navigation guards are used). Possibly, such obtained navigation steps that belong to a composite analysis situation are never invoked due to a navigation guard that is always evaluated to false.

¹⁵Because a surjective graph homomorphism only claims that there is a surjective mapping to the set of vertices, we extend the condition that also all navigation step schemas have to be instantiated (note, we allow to have multi-graphs at schema level).

schema, an instance is included in the composite analysis situation meaning that the composite analysis situation represents an instance of the whole composite analysis situation schema.¹⁶

In the definition of a composite analysis situation schema, it is possible that one or more navigation step schemas of a composite analysis situation schema end up to an analysis situation schema that already has been instantiated. Such a sequence of navigation step schemas represents a loop at schema level. Moreover, there are loops of navigation step schemas that generate analysis situations that have been instantiated previously by the loop. As a simplified example, consider navigation step schemas $NAV_1 = (AS1, \text{true}, \text{drillDownOneLevel}(D), (\text{Insurant}), AS2)$ and $NAV_2 = (AS2, \text{true}, \text{rollUpOneLevel}(D), (\text{Insurant}), AS1)$. There is a navigation step schema from analysis situation schema $AS1$ to analysis situation step schema $AS2$ and, conversely, from $AS2$ to $AS1$. Considering such loops, it is possible that the instantiation of a analysis situation schema creates the same analysis situation that already has been instantiated previously. For example, if analysis situation $as1$ is an instance of analysis situation schema $AS1$, navigation step $as2 = as1.\text{drillDownOneLevel}(\text{Insurant})$ represents an instance of navigation step schema NAV_1 and navigation step $as1 = as2.\text{rollUpOneLevel}(\text{Insurant})$ represents an instance of navigation step schema NAV_2 that again ends up to analysis situation $as1$. In this case, infinite sequences of instantiations could arise—in our example, after the second instantiation of analysis situation $as1$, analysis situation $as2$ can be instantiated a second time via navigation step schema NAV_1 , afterwards, analysis situation $as1$ can be instantiated a third time, etc. For composite analysis situation schemas, the instantiation process has to be terminated to avoid infinite instantiation loops with respect to an automatic instantiation process (instantiation in one go) necessary for composite analysis situation schemas. Thus, by the third condition, Definition 6.21 requires explicitly that, for an analysis situation schema of a composite analysis situation schema, an analysis situation cannot be instantiated twice.

¹⁶Note again that navigation steps of such a composite analysis situation are not invoked, if they comprise a navigation guard that is evaluated to false, i.e., possibly, an analysis situation is included in a composite analysis situation but it is not used for query execution because navigation is aborted at instance level due to a navigation guard.

This ensures a deterministic instantiation of a composite analysis situation schema that do not contain infinite sequences of analysis situations.

The fourth condition of Definition 6.21 ensures that the maximal number of navigation steps are instantiated when a composite analysis situation schema is instantiated. Condition 2 already assures that each navigation step schema of a composite analysis situation schema is used in the instantiation process. If there are loops in a composite analysis situation schema, a navigation step schema of the composite analysis situation schema can be used for instantiation several times. Condition 4 specifies that a maximal number of navigation steps is generated starting from the root. This also ensures that the whole composite analysis situation schema is instantiated in a unique way (deterministic instantiation process). If a navigation step \mathbf{nav} is an element of the composite analysis situation \mathbf{as} and if a navigation step schema NAV' allows to append another navigation step \mathbf{nav}' , then this navigation step again is an element of composite analysis situation \mathbf{as} . The only exception represents the case that such a navigation step creates an analysis situation that already exists with respect to the instantiation process. In this case, infinite instantiation loops have to be prevented.

Finally, it has to be mentioned explicitly that the concept of composite analysis situation schemas and composite analysis situations can be used in a hierarchical manner. A composite analysis situation schema is a subgraph of a BI analysis graph schema. This subgraph again represents another BI analysis graph schema that can comprise other composite analysis situation schemas. Hence, composite analysis situation schemas can also be defined hierarchically in the sense that composite analysis situation schemas can also contain other composite analysis situation schemas. Analogously, at instance level, a composite analysis situation can also contain other composite analysis situations.

6.5.4 Graphical Representation and Examples

In this subsection, the graphical representation and examples of composite analysis situation schemas and composite analysis situations are presented.

Graphically, on one side, composite analysis situation schemas and composite analysis situations have to represent analysis situation schemas and analysis situations, respectively, on the other side, they also have to be considered as constructs that represent subgraphs of BI analysis graph schemas and BI analysis graphs. Furthermore, as non-comparative and comparative analysis situation schemas and analysis situations can be represented in full, lean, and condensed graphical representations, a suitable visualization for composite analysis situation schemas and composite analysis situations is chosen that also can be easily adopted for all three kinds of graphical representations. We introduce the graphical representation of composite analysis situation schemas and composite analysis situations by examples and focus on lean graphical representations.

Whereas composite analysis situation schemas are drawn with a double-edged border, composite analysis situations are depicted with a single-edged one (accordingly to non-comparative and comparative analysis situation schemas and analysis situations). In Figure 6.20, a composite analysis situation schema is presented and Figure 6.22 shows two composite analysis situations that represent two instances of this composite analysis situation schema. Composite analysis situation schemas as well as composite analysis situations comprise a header and a lower part that include the subgraph which defines a composite analysis situation schema and a composite analysis situation, respectively. The header contains a specific pictogram (symbolizing a composite analysis situation schema and a composite analysis situation, respectively), a name (identifying the composite analysis situation schema and composite analysis situation), and an optional description (analogously to non-comparative and comparative analysis situations schemas and analysis situations—but not depicted in the examples of this subsection). This graphical representation can be considered as a lean graphical representation. In a condensed graphical representation, only the header of a composite analysis situation schema and composite analysis situation is drawn. Again, the defining components (components of the subgraph) are hidden as in the condensed graphical notation of non-comparative and comparative analysis situation schemas and analysis situations. A full graphical representation

of composite analysis situation schemas and composite analysis situations is drawn in the same style and with the same content like in the lean notation except that the borders are not drawn as rounded but as common rectangles comprising sharp corners. The examples in this subsection do not demonstrate full graphical representations of composite analysis situation schemas and composite analysis situations.

Figure 6.20 presents a composite analysis situation schema with name *CompositeAS-4* comprising analysis situation schemas *AS-4*, *AS-5*, and *AS-7*, and navigation step schemas in $AS-4 \Rightarrow AS-5$ and $AS-4 \Rightarrow AS-7$. Analysis situation schema *AS-4* represents the root of composite analysis situation schema *CompositeAS-4*. The root is additionally marked by a white square in the upper right corner. Although the incoming navigation step schema in $AS-3 \Rightarrow AS-4$ would signify analysis situation schema *AS-4* as the root, a specific pictogram for identifying a root makes sense. Especially in the case of composite analysis situation schemas that do not have incoming navigation step schemas (which is allowed by the definition), it is indispensable to highlight the root graphically. Note that all navigation step schemas that belong to the set of input navigation step schemas of the composite analysis situation schema must have root *AS-4* as target analysis situation schema (the unique input analysis situation schema). Graphically, each input navigation step schema to the root is additionally marked by a small white rectangle with an optional label (in our example, the small white rectangle is labelled by name *in*). There are two navigation step schemas that belong to the set of output navigation step schemas of composite analysis situation schema *CompositeAS-4*. The first one belongs to set $AS-5 \Rightarrow AS-6$ and the other one to set $AS-7 \Rightarrow AS-8$. The connection to the source is marked by a small greyed colored square. In the example of Figure 6.20, the grey colored square of the first navigation step schema is labelled by name *out_{rural}* and the grey colored square of the second navigation step schema is labelled by name *out_{urban}*. The specific graphical marking and labelling of the connection of the input and output navigation step schemas represents only a graphical decoration symbolizing the transition from the environment to the composite analysis situation schema. These graphical components are not part of

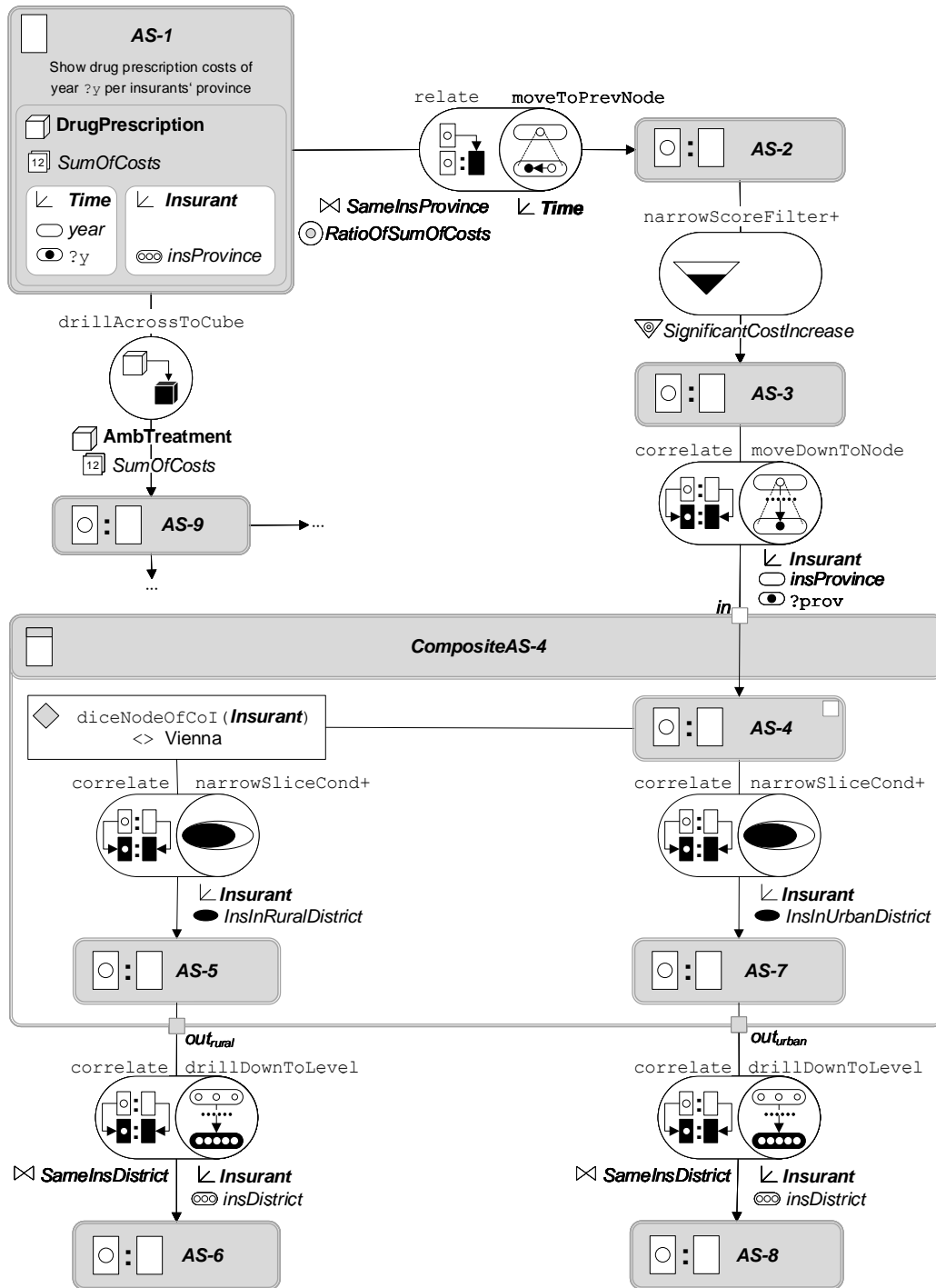


Figure 6.20: Running example presented in Figure 6.2 including a composite analysis situation schema

the formal definition of a composite analysis situation schema.

A condensed graphical representation of the example of Figure 6.20 is depicted in Figure 6.21. Input label *in* and output labels *out_{rural}* and *out_{urban}* are again only decorations in the graphical representation. The composite analysis situation schema is only represented by the header containing the name of the composite analysis situation schema and its pictogram. As in general, also this condensed depiction does not show the internal structure of the composite analysis situation schema. Hence, the target analysis situation schema of the input navigation step schema and the sources of both output navigation step schemas are not visible.¹⁷ Analogously to schema level, a composite analysis situation can be depicted in condensed graphical representation at instance level.

In Figure 6.22, an instance of analysis graph schema presented in Figure 6.20 is demonstrated in lean graphical representation. This example comprises two composite analysis situations (**compositeAS-4.1** and **compositeAS-4.2**) which are instances of composite analysis situation schema *CompositeAS-4* depicted in Figure 6.20. The root analysis situation of composite analysis situation **compositeAS-4.1** is named as **as-4.1** and the root of composite analysis situation **compositeAS-4.2** is named as **as-4.2**, both visually marked by a white square in the upper right corner (like at schema level). Both roots (analysis situation **as-4.1** and analysis situation **as-4.2**) are instances of analysis situation schema *AS-4*. Analogously to schema level, at instance level, navigation steps that are instances of input navigation step schemas of a composite analysis situation schema are marked by a white labelled square and navigation steps that are instances of output navigation step schemas of a composite analysis situation schema are marked by a grey labelled square. The composite analysis situations of Figure 6.22 comprises two navigation steps (one with target analysis situation **as-4.1** and the other one with target analysis situation **as-4.2**) that are marked by a white square and labelled by name *in* (both labels transferred from the schema level) and three navigation steps marked by a grey square and labelled by the names *out_{rural}* and *out_{urban}*

¹⁷If tool support is provided, one can envision that by a user's mouse click, the subgraph of a composite analysis situation schema becomes visible.

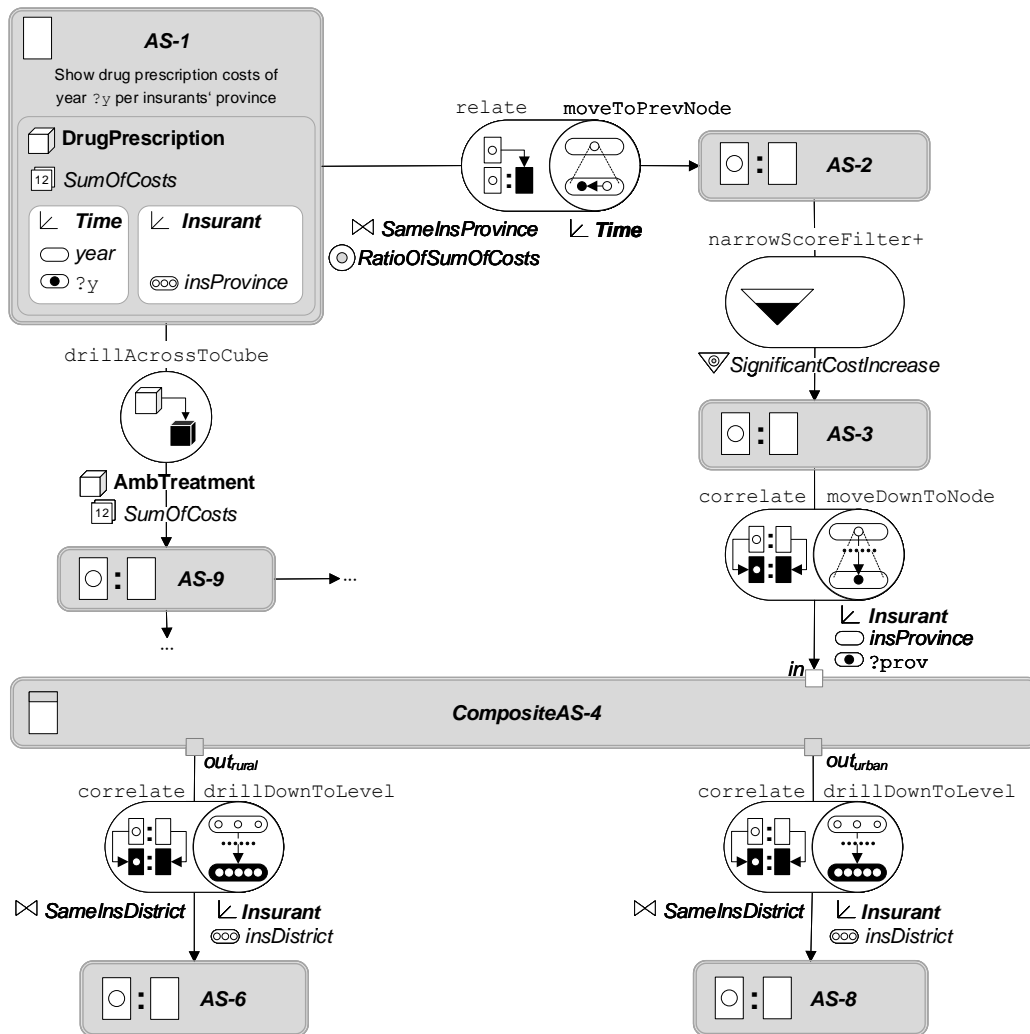


Figure 6.21: Example of Figure 6.20 in condensed graphical representation

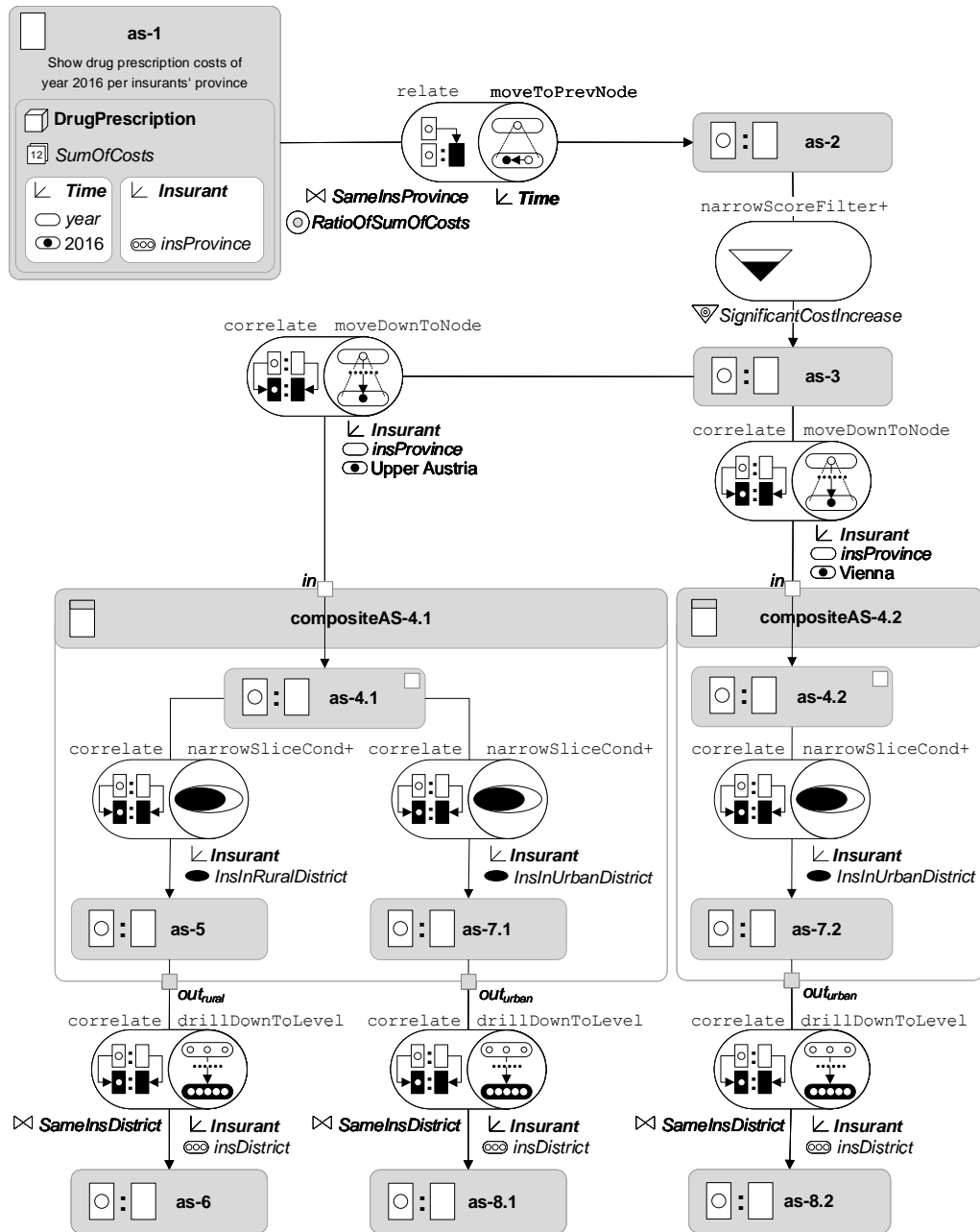


Figure 6.22: Composite analysis situations instantiated from composite analysis situation schema presented in Figure 6.20

(again transferred from the schema level). Note, in Figure 6.22, two navigation steps go out from composite analysis situation `compositeAS-4.1` which are labelled by names `outrural` and `outurban`, and one navigation step goes out from composite analysis situation `compositeAS-4.2` which is labelled by name `outurban`. Formally, another navigation step of `compositeAS-4.2` that narrows to insurants' rural districts is instantiated from composite analysis situation schema `CompositeAS-4`. But in this case (province Vienna), the navigation guard is always evaluated to false. Thus, this navigation step is not depicted in Figure 6.22.

Figure 6.20 shows an example of a composite analysis situation schema that can be used to instantiate composite analysis situations to obtain information about drug prescription costs of a certain province as a whole (analysis situation schema `AS-4`) and, additionally, narrowed to rural (analysis situation schema `AS-5`) and urban (analysis situation schema `AS-7`) districts. Note, the information generated by the queries of the instances of these three analysis situation schemas is obtained in one go. By the instantiation of root `AS-4`, also analysis situation schemas `AS-5` and `AS-7` are instantiated and the corresponding queries are executed.¹⁸ If one wants to obtain additional lists containing drug prescription costs per district (one list for rural and the other one for urban districts), she or he can navigate explicitly to an instance of analysis situation schema `AS-6` and to an instance of analysis situation schema `AS-8`.¹⁹

It depends on the intention of the business analyst how analysis situations are conflated into composite analysis situations. Figure 6.23 shows the same example of a BI analysis graph schema except that there are different assemblies of analysis situation schemas which form composite analysis situation schemas. Composite analysis situation schema `CompositeAS-5` consists of analysis situation schemas `AS-5` and `AS-6`, and composite analysis situation schema `CompositeAS-7` comprises analysis situation schemas `AS-7` and `AS-8`.

¹⁸One can say that these three queries are created and executed together automatically without further user interaction.

¹⁹One can say that this query creation and execution is not performed automatically but requires additional user interaction because analysis situation schemas `AS-6` and `AS-8` are not part of composite analysis situation schema `CompositeAS-4`.

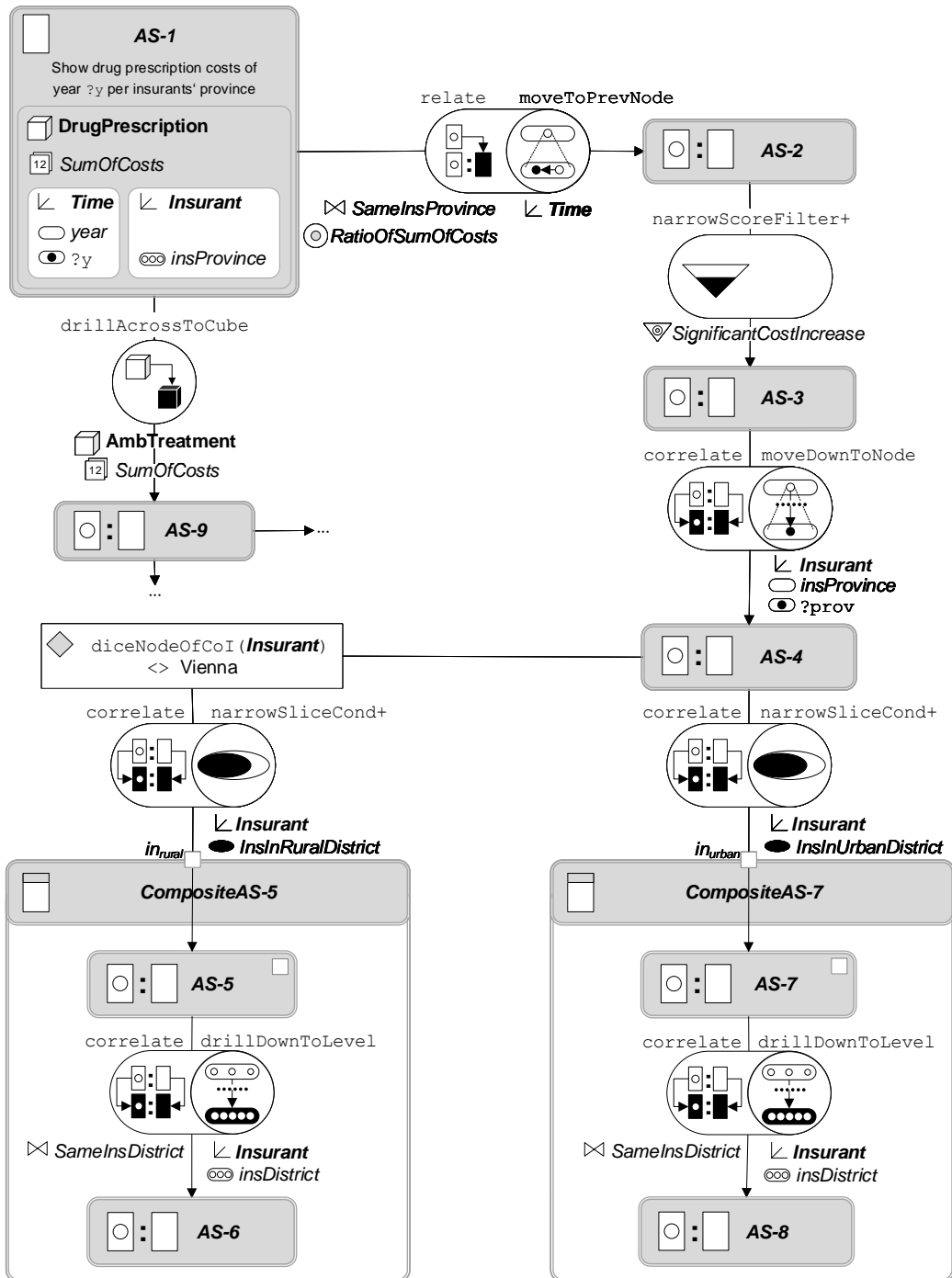


Figure 6.23: The same example as presented in Figure 6.20 except another assemblies of analysis situation schemas to composite analysis situation schemas

In this example, the drill-down navigation step schemas to districts are encapsulated in composite analysis situation schemas. Whereas composite analysis situation schema *CompositeAS-5* represents an automatic drill-down to rural districts, composite analysis situation schema *CompositeAS-7* contains an automatic drill-down to urban districts. Analysis situation schema *AS-5* is the root of composite analysis situation schema *CompositeAS-5* and analysis situation schema *AS-7* is the root of composite analysis situation schema *CompositeAS-7*. The input navigation step schema of composite analysis situation schema *CompositeAS-5* is labelled by name *in_{rural}* and input navigation step schema of composite analysis situation schema *CompositeAS-7* is labelled by name *in_{urban}*. In both composite analysis situation schemas, there are no output navigation step schemas. If root *AS-5* is instantiated, also analysis situation *AS-6* is instantiated and both resulting queries are executed. Analogously, if root analysis situation schema *AS-7* is instantiated, also analysis situation schema *AS-8* is instantiated and, again, both resulting queries are executed.

Figure 6.24 presents an instance of BI analysis graph schema of Figure 6.23. Composite analysis situation *compositeAS-5* represents an instance of composite analysis situation schema *CompositeAS-5* and composite analysis situations *compositeAS-7.1* and *compositeAS-7.2* are instances of composite analysis situation schema *CompositeAS-7*. Analysis situations *as-5*, *as-7.1*, and *as-7.2* represent the corresponding roots. Accordingly to the schema level, the input navigation steps are labelled by names *in_{rural}* and *in_{urban}*. Analogously to the schema level, there are no output navigation steps at instance level.

A composite analysis situation schema having two input navigation step schemas is demonstrated in Figure 6.25. We compare this example with the composite analysis situation schemas of Figure 6.23. In Figure 6.23, two composite analysis situation schemas are depicted. Composite analysis situation schema *CompositeAS-5* comprises analysis of rural districts and composite analysis situation schema *CompositeAS-7* contains analysis of urban districts. The root analysis situation schemas are the targets of navigation step schemas that narrow the slice conditions to rural and urban districts, respec-

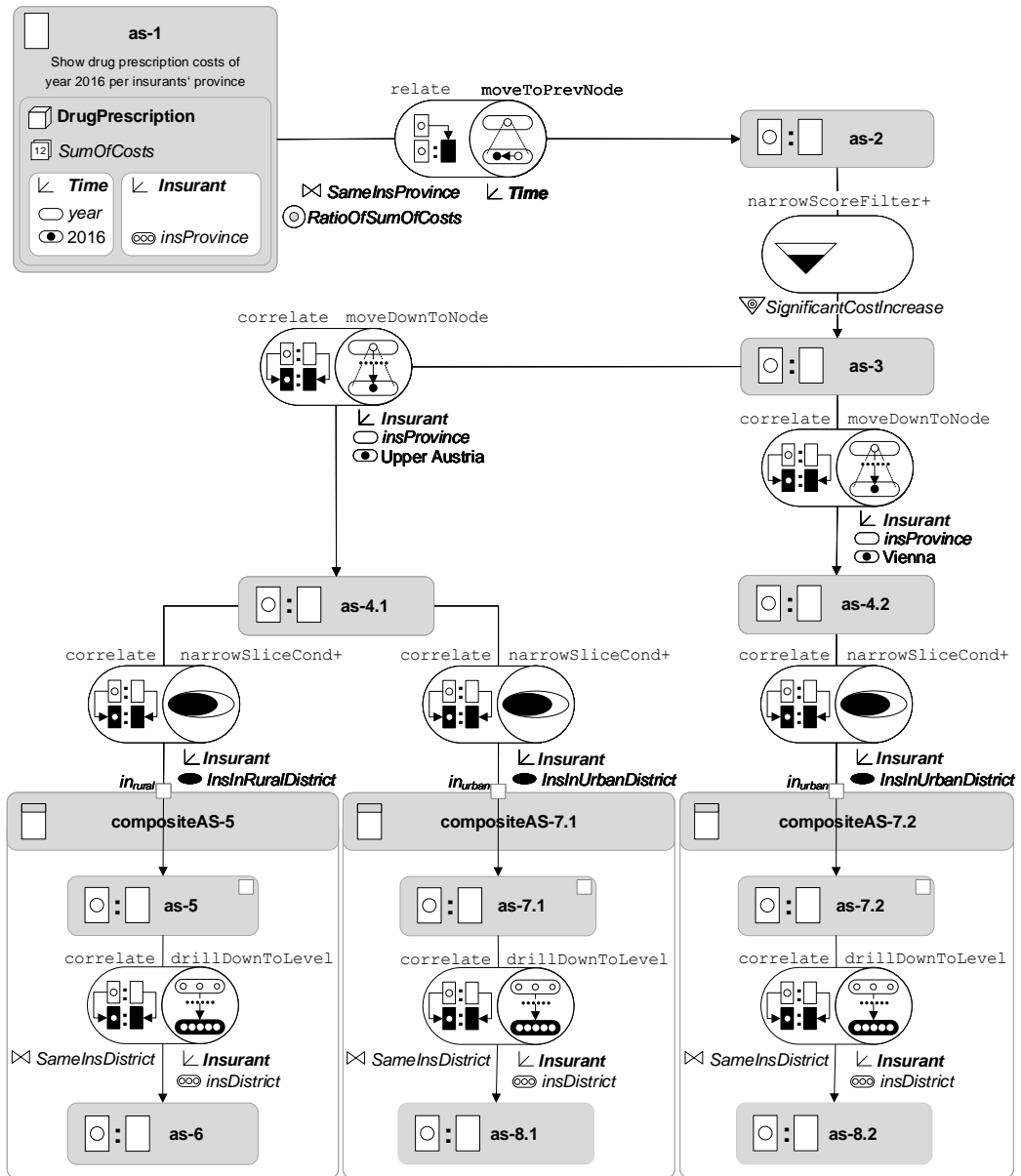


Figure 6.24: Composite analysis situations instantiated from composite analysis situation schemas presented in Figure 6.23

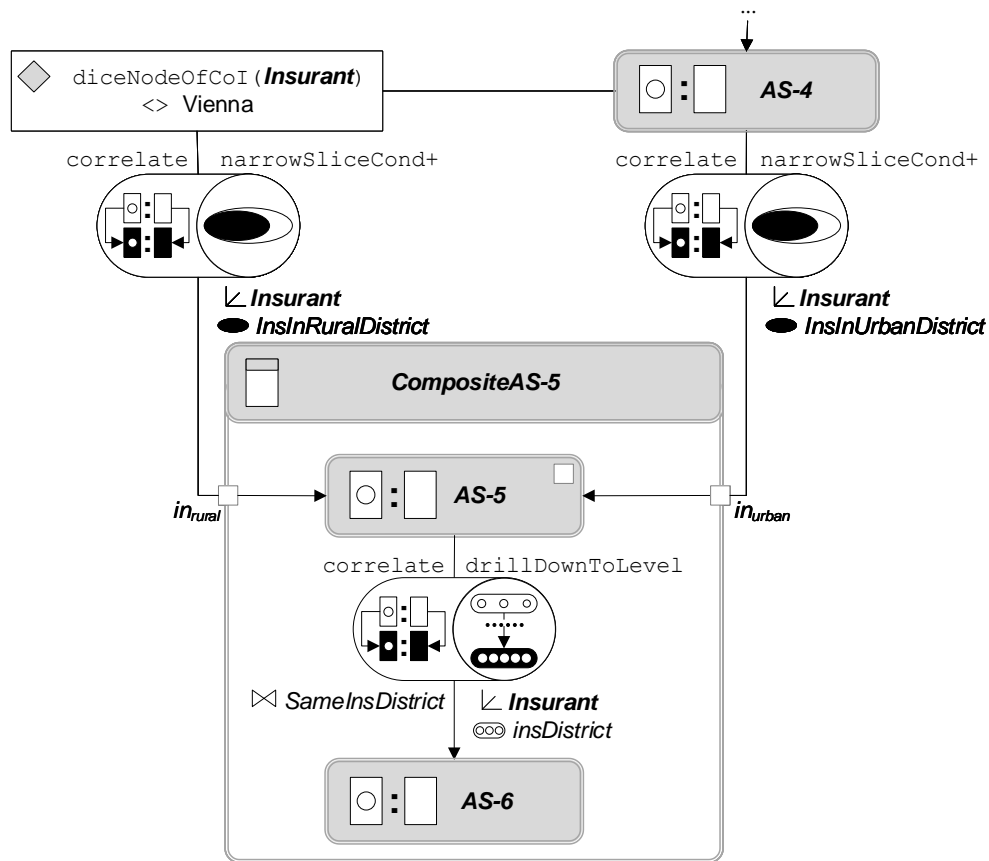


Figure 6.25: Alternative modeling of the composite analysis situation schemas presented in Figure 6.23

tively. If one introduces variables for slice conditions, both composite analysis situation schemas can be reduced to one. Figure 6.25 shows such a composite analysis situation schema having two input navigation step schemas with a mandatory common root. In this example, we assume to have the same BI analysis graph schema as demonstrated in Figure 6.23 except that there is only one composite analysis situation schema *CompositeAS-5* comprising analysis situation schemas *AS-5* and *AS-6* instead of two composite analysis situation schemas *CompositeAS-5* and *CompositeAS-7* comprising *AS-5* and *AS-6*, and *AS-7* and *AS-8*, respectively, as depicted in Figure 6.23. Moreover, in Figure 6.25, we assume to have variables for the slice conditions (of the context of interest and the context of comparison) in analysis situation schemas *AS-5* and *AS-6*.²⁰ There are two input navigation step schemas that lead to composite analysis situation schema *CompositeAS-5*. In this example, both navigation step schemas have analysis situation schema *AS-4* as source. On the other side, root analysis situation schema *AS-5* is the mandatory common target of both input navigation step schemas.²¹

The composite analysis situations *compositeAS-5*, *compositeAS-7.1*, and *compositeAS-7.2* presented in Figure 6.24 can also be considered as instances of composite analysis situation schema *CompositeAS-5* of Figure 6.25. In this case, the variables for the slice conditions are bound to dimensional predicates *InsInRuralDistrict* and *InsInUrbanDistrict*, respectively, when the input navigation step schemas of composite analysis situation schema *CompositeAS-5* are instantiated.

In Figure 6.26, an example of a composite analysis situation schema with name *CompositeAS* is presented that comprises a navigation step schema (the single navigation step schema of set $AS-3 \Rightarrow AS-3$) which can be considered as a loop for instantiation. There is one input navigation step schema

²⁰Note, in condensed graphical notation, the variables of the slice conditions are not visible as it would be in lean or full graphical notation. Thus, in this example, this fact for analysis situation schemas *AS-5* and *AS-6* is mentioned in the running text as an assumption. The condensed graphical notation was chosen to obtain a smaller picture.

²¹Note, it is not necessary that two input navigation step schemas to a composite analysis situation schema have the same source but, indeed, it is necessary that two input navigation step schemas of a composite navigation step schema have the same target which represents the root of the composite analysis situation schema.

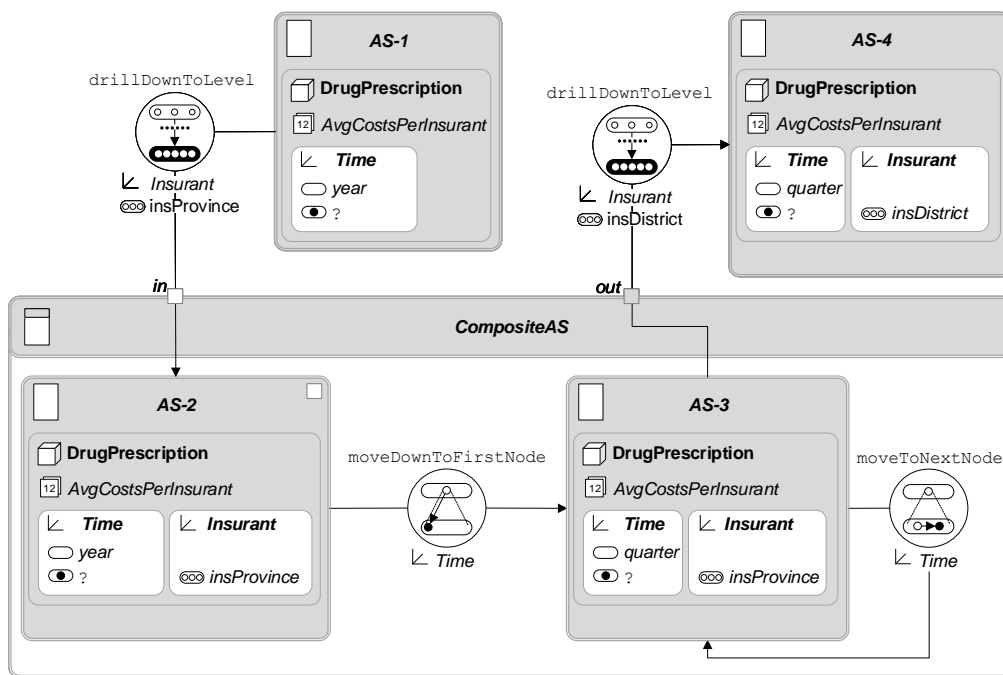


Figure 6.26: Example of a composite analysis situation schema containing a loop

from analysis situation schema *AS-1* to analysis situation schema *AS-2* decorated with label *in* and one output navigation step schema from analysis situation schema *AS-3* to analysis situation schema *AS-4* fitted with label *out*. Composite analysis situation schema *CompositeAS* comprises analysis situation schemas *AS-2* and *AS-3* and a navigation step schema from analysis situation schema *AS-2* to analysis situation schema *AS-3*, and another navigation step schema from analysis situation schema *AS-3* to it itself. Analysis situation schema *AS-2* represents the root of composite analysis situation schema *CompositeAS*. In this example of a BI analysis graph schema, analysis situation schema *AS-1* is used to instantiate analysis situations that calculate average drug prescription costs per insurant with respect to a certain year which is represented by a variable that has to be bound at instantiation time. The navigation step schema to analysis situation schema *AS-2* drills down to insurants' provinces. This navigation step schema leads to composite analysis situation schema *CompositeAS*. An instance of analysis situation schema *AS-2* automatically induces an instance of analysis situation schema *AS-3* which moves down to the first quarter of the corresponding year. Analogously, navigation step schema from analysis situation schema *AS-3* to itself creates analysis situations for subsequent quarters of the same year (by navigation operator `moveToNextNode`) until the operator's preconditions are not satisfied any more which is the case, if the last quarter of the corresponding year has already been reached. The navigation step schema with source analysis situation schema *AS-3* and target analysis situation schema *AS-4* specifies a drill-down to insurants' districts. Analysis situation schema *AS-4* does not belong to composite analysis situation schema *CompositeAS*. Thus, this analysis situation schema would not be instantiated automatically when the composite analysis situation is instantiated. One can consider this navigation step schema as a drill-down to insurants' districts which must be explicitly induced by the user.

An instance of BI analysis graph schema of Figure 6.26 is depicted in Figure 6.27. One can consider that a business analyst binds the free variable of analysis situation schema *AS-1* to year 2016 and navigates to analysis situation schema *AS-2* by drilling down to the insurants' provinces—analysis situ-

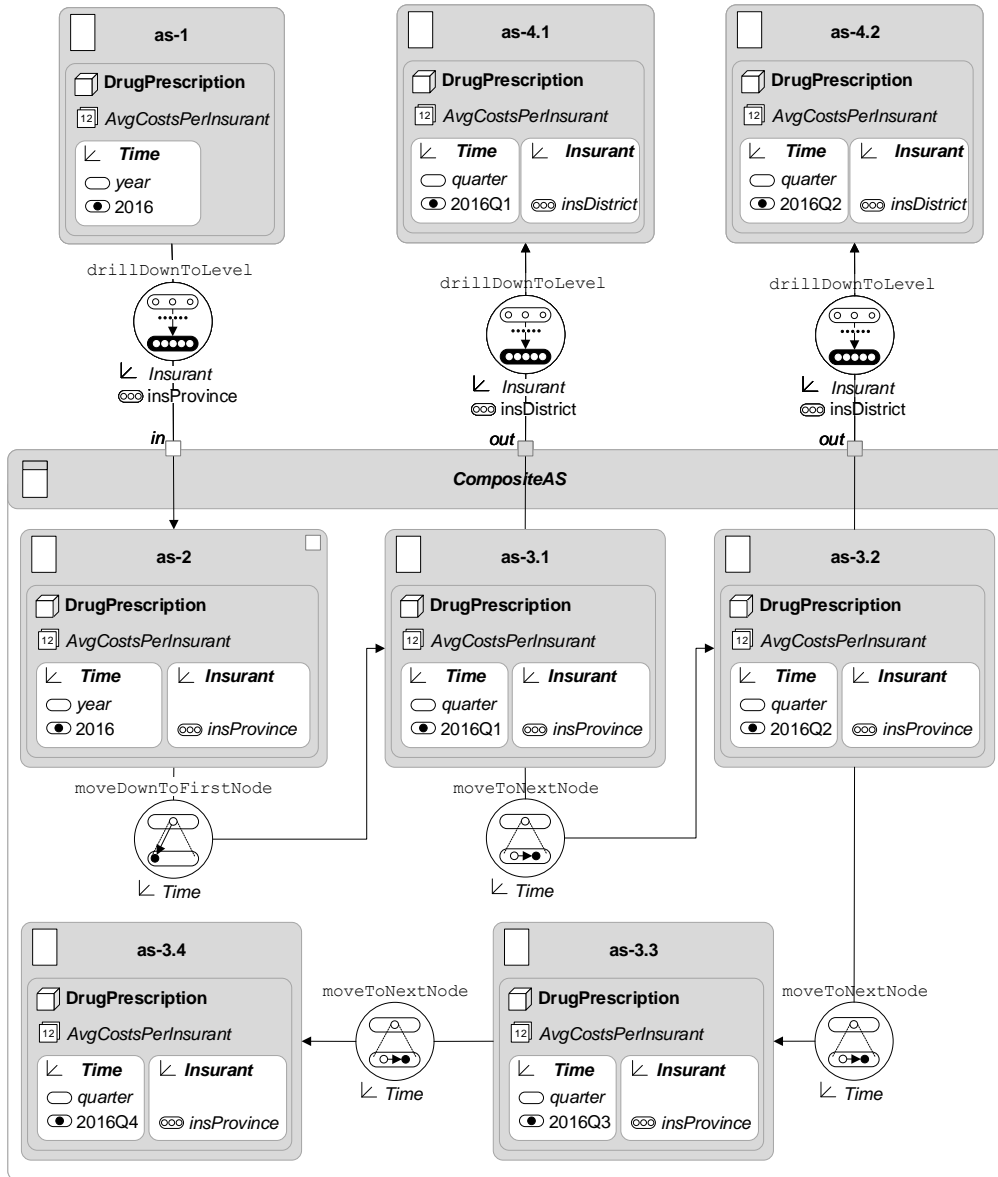


Figure 6.27: A composite analysis situation instantiated from composite analysis situation schema presented in Figure 6.26

ations *as-1* and *as-2* are generated. By instantiating analysis situation schema *AS-2* (the root of composite analysis situation schema *CompositeAS*), navigation step schemas of $AS-2 \Rightarrow AS-3$ and $AS-3 \Rightarrow AS-3$ are instantiated automatically, and all possible navigation steps (with respect to the navigation step schemas included in the composite analysis situation schema) are created till the precondition of a potential navigation step evaluates to false. These instantiations are performed automatically such that analysis situations *as-2*, *as-3.1*, *as-3.2*, *as-3.3*, and *as-3.4* are generated in one go.²² The execution of analysis situation *as-2* returns average drug prescription costs of year 2016, and the executions of analysis situations *as-3.1*, *as-3.2*, *as-3.3*, and *as-3.4* return average drug prescription costs of quarters 2016Q1, 2016Q2, 2016Q3, and 2016Q4. For quarters 2016Q1 and 2016Q2, the user explicitly performs a drill-down to dimension level *insDistrict* such that analysis situations *as-4.1* and *as-4.2* are created. Analysis situation schema *AS-4* does not belong to composite analysis situation schema *CompositeAS*. Thus, both instance *as-4.1* and instance *as-4.2* are not created automatically but explicitly by the user. In contrast, the user is not interested in drilling down to insurers' provinces for quarters 2016Q3 and 2016Q4. Hence, no appropriate instantiations were performed for these quarters.

6.6 Analysis Trace and Backtracking

A BI analysis graph is a directed tree that contains analysis situations which are connected by navigation steps. Analysis situations represent queries and a sequence of navigation steps defines a temporal order of analysis situations, i.e., it specifies when a query (source) is performed before another one (target). Although we have a temporal order along navigation steps that yields a path consisting of consecutive navigation steps, we have no one

²²Note that navigation step schema of set $AS-2 \Rightarrow AS-3$ is used to instantiate analysis situation *as-3.1* and navigation step schema of set $AS-3 \Rightarrow AS-3$ is used three times to instantiate analysis situations *as-3.2*, *as-3.3*, and *as-3.4*. Condition 2 of Definition 6.21 ensures that navigation step schema of set $AS-3 \Rightarrow AS-3$ is used for instantiation generally and condition 4 ensures that this navigation step schema is used for instantiation two more times until the navigation operator's preconditions does not hold any more.

between different branches that start from a source analysis situation and lead to different subtrees²³. One also could think of parallel (or concurrent) execution of queries (or parallel instantiation of analysis situations) that belong to different subtrees yielding from different branches. This parallelism can be interpreted as a parallel execution plan of queries but it also can be considered as alternative options within the analysis process.

The following definition formalizes a parallel analysis trace of a non-empty BI analysis graph. It is just another formalism of the underlying directed tree except preserving the operation performed by a navigation step, i.e., a parallel analysis trace also represents a BI analysis graph but without the corresponding navigation steps. Only analysis situations and the order that defines which analysis situation follows another analysis situation are represented. The semantic difference expressed by a navigation operator (and its actual parameters) is not included in a parallel analysis trace.

Definition 6.22. A *parallel analysis trace* $PAT = (\mathbf{r}, T)$ of a non-empty BI analysis graph \mathbf{ag} is recursively defined in the following way:

1. $\mathbf{r} = \text{Root}_{\mathbf{ag}}$ and
2. T is an empty set ($T = \emptyset$), if \mathbf{ag} has no subtrees, or, T is a non-empty set $T = \{T_1, \dots, T_n\}$, if the whole BI analysis graph \mathbf{ag} consists of n (with $n \geq 1$) subtrees $\mathbf{sg}_1 \dots \mathbf{sg}_n$, where, for $1 \leq i \leq n$, T_i represents a parallel analysis trace of subtree \mathbf{sg}_i .

Moreover, the root \mathbf{r} of parallel analysis trace PAT is denoted by Root_{PAT} . We also use the following syntax to present a parallel analysis trace: $id_{\mathbf{r}} \rightarrow (T_1 | \dots | T_n)$, in the case of $PAT = (\mathbf{r}, \{T_1, \dots, T_n\})$ with $n > 1$, $id_{\mathbf{r}} \rightarrow \text{Root}_{T_1}$, in the case of $PAT = (\mathbf{r}, \{T_1\})$, and $id_{\mathbf{r}}$, in the case of $PAT = (\mathbf{r}, \emptyset)$. □

Definition 6.22 reflects the tree structure of a BI analysis graph without navigation operators. Formally, this tree structure is represented as a pair comprising the root of the tree and a set containing all subtrees. In a recursive

²³A subtree of a tree comprises a node (the root of the subtree) and all its descendent nodes of the encompassing tree.

way, these subtrees are considered again as parallel analysis traces. The leaves of the tree have the same pair structure except that the set of subtrees is empty.

All subtrees of the root of a BI analysis graph are collected by a set and not by a sequence. Thus, no temporal ordering between subtrees can be deduced or, in other words, all subtrees have to be considered parallel in a temporal view. Note, in contrast, there exists a temporal order from the root to the roots of the subtrees in the sense that the root analysis situation must be executed before the root analysis situations of the subtrees are executed. The mentioned parallelism refers the temporal order between the roots of the subtrees, i.e., it is not defined which root analysis situation of the subtrees has to be executed first and which one later.

In the definition, another syntactical notation is introduced. In this notation, analysis situations are indicated by their identifiers. Arrow \rightarrow symbolizes a sequential step from the root to a subtree or, to be precise, to the root of a subtree, whereas, symbol $|$ associates parallelism between subtrees. All parallel subtrees are syntactically embraced by parentheses. Analogously to sets, the syntactical notation with symbol $|$ does not claim an order, i.e., the syntactical order can be changed and, semantically, the parallel analysis traces still remains the same. Leaves are only represented by the identifier of the root (the empty set of subtrees is omitted syntactically) and, in the case that there is only one subtree, the embracing parentheses are also omitted.

The BI analysis graph of Figure 6.1 can be represented by the following parallel analysis trace: $(\text{as-1}, \{ (\text{as-2}, \{ (\text{as-3}, \{ (\text{as-4.1}, \{ (\text{as-5}, \{ (\text{as-6}, \emptyset) \}) , (\text{as-7.1}, \{ (\text{as-8.1}, \emptyset) \}) \}) \}) , (\text{as-4.2}, \{ (\text{as-7.2}, \{ (\text{as-8.2}, \emptyset) \}) \}) \}) \}) \})$. This parallel analysis trace can also be expressed in the following syntactically more convenient notation which is better readable: $\text{as-1} \rightarrow \text{as-2} \rightarrow \text{as-3} \rightarrow (\text{as-4.1} \rightarrow (\text{as-5} \rightarrow \text{as-6} \mid \text{as-7.1} \rightarrow \text{as-8.1}) \mid \text{as-4.2} \rightarrow \text{as-7.2} \rightarrow \text{as-8.2})$. One can identify the branching at analysis situations **as-3** and **as-4.1**. In both cases, analysis situations **as-3** and **as-4.1** represent the roots and both comprise two subtrees that are separated by symbol $|$ and that are embraced by parentheses.

A parallel analysis trace of a BI analysis graph associates concurrency

that can be applied, if, for example, the execution of BI analysis graphs is automated (concurrent execution in runtime environments) or if the query results can be analyzed by more than one business analysts in parallel (labor division). If there is a need for sequencing all analysis situations of a BI analysis graph, one has to select an appropriate tree traversal. For automation, depth-first or breath-first traversals are easy to implement. A human business analyst would apply both traversal strategies often in a mixed way depending on her or his experience and heuristic.

The following definition of a sequential analysis trace does not specify a specific traversal strategy (for instance, depth-first or breath-first traversal) but it expresses a sequencing of analysis situations of a BI analysis graph in a temporal order where additional conditions have to be respected. Such a sequence has to comprise all analysis situations of a BI analysis graph and it has to respect the navigation steps in the sense that two consecutive analysis situations of the sequence are founded by a navigation step except backtracking steps that jump back to an analysis situation that already was executed previously. Backtracking is necessary to jump back to previous analysis situations to navigate through further branches and subtrees of a BI analysis graph.

Definition 6.23. A *sequential analysis trace* (or simply also denoted as *analysis trace*) $SAT = (\mathbf{as}_1, \dots, \mathbf{as}_n)$ of a non-empty BI analysis graph \mathbf{ag} is defined as a sequence of analysis situations that satisfies the following conditions:

1. $\{\mathbf{as}_i \mid 1 \leq i \leq n\} = ASituations(\mathbf{ag})$,
2. $\mathbf{as}_1 = Root_{\mathbf{ag}}$ and,
3. for $1 \leq i \leq n - 1$,
 - (a) there is a navigation step from \mathbf{as}_i to \mathbf{as}_{i+1} in BI analysis graph \mathbf{ag} such that, for $1 \leq j \leq n - 1$ with $j \neq i$, $\mathbf{as}_i \neq \mathbf{as}_j$ and $\mathbf{as}_{i+1} \neq \mathbf{as}_{j+1}$, or,
 - (b) if $2 \leq i \leq n - 2$ and if there is no navigation step from \mathbf{as}_i to \mathbf{as}_{i+1} in BI analysis graph \mathbf{ag} , then there exists an index j with

$1 \leq j < i$ such that $\mathbf{as}_{i+1} = \mathbf{as}_j$ and there are navigation steps in BI analysis graph \mathbf{ag} from \mathbf{as}_{i-1} to \mathbf{as}_i as well as from \mathbf{as}_{i+1} to \mathbf{as}_{i+2} .

In case 3a, we also use notation $id_{\mathbf{as}_i} \rightarrow id_{\mathbf{as}_{i+1}}$ and, in case 3b, notation $id_{\mathbf{as}_i} \hookrightarrow id_{\mathbf{as}_{i+1}}$ is used and this case is also referred to as *backtracking step*. \square

Condition 1 of Definition 6.23 ensures that all analysis situations of BI analysis graph \mathbf{ag} are included in the sequential analysis trace and, by condition 2, the first element of sequence SAT represents the root of \mathbf{ag} . The third condition comprises two cases that interpret the steps from a sequence element to the subsequent one. Condition 3a refers to navigation steps of BI analysis graph \mathbf{ag} . Note, similarly to parallel analysis traces, navigation steps themselves are not included in a sequential analysis trace but they induce that the source and target of a navigation step occur as two consecutive sequence elements. Together with condition 1, one can conclude that each navigation step of the BI analysis graph induces a pair of consecutive sequence elements. Additionally, condition 3a expresses that such a pair is unique, i.e., a navigation step induces exactly one unique pair of consecutive sequence elements. In the second case represented by condition 3b, two consecutive sequence elements define a backtracking step which means that from an analysis situation \mathbf{as}_i there is a transition to analysis situation \mathbf{as}_{i+1} that already was a sequence element at a previous position (in Definition 6.23 denoted as \mathbf{as}_j). The only purpose of such backtracking steps is to perform directly a navigation step from a source analysis situation which already was executed at a previous trace position to a target analysis situation that was not yet executed. Thus, a backtracking step may not be the first or last step in the sequential analysis trace and a backtracking step may not be directly preceded or directly followed by another backtracking step. These requirements are also included in condition 3b.

Syntactically, we use two different arrows to express the two cases of condition 3 in Definition 6.23. Arrow \rightarrow represents an induced navigation step and arrow \hookrightarrow represents a backtracking step. Both types of arrows are used for an alternative sequence representation that define a sequential

temporal order. The specific type of arrows for backtracking step allows to recognize such steps more quickly.

Considering the example of Figure 6.1, a sequential analysis trace can be specified following, for example, by a depth-first traversal: (as-1, as-2, as-3, as-4.1, as-5, as-6, as-4.1, as-7.1, as-8.1, as-3, as-4.2, as-7.2, as-8.2). In arrow notation, the same sequence is written as $as-1 \rightarrow as-2 \rightarrow as-3 \rightarrow as-4.1 \rightarrow as-5 \rightarrow as-6 \hookrightarrow as-4.1 \rightarrow as-7.1 \rightarrow as-8.1 \hookrightarrow as-3 \rightarrow as-4.2 \rightarrow as-7.2 \rightarrow as-8.2$. The arrow notation allows to perceive backtracking steps ($as-6 \hookrightarrow as-4.1$ and $as-8.1 \hookrightarrow as-3$) more quickly.

Note that this sequential analysis trace of the BI analysis graph of Figure 6.1 is not the only one. For example, one can define another sequential analysis trace that is based on breath-first traversal: $as-1 \rightarrow as-2 \rightarrow as-3 \rightarrow as-4.1 \hookrightarrow as-3 \rightarrow as-4.2 \hookrightarrow as-4.1 \rightarrow as-5 \hookrightarrow as-4.1 \rightarrow as-7.1 \hookrightarrow as-4.2 \rightarrow as-7.2 \hookrightarrow as-5 \rightarrow as-6 \hookrightarrow as-7.1 \rightarrow as-8.1 \hookrightarrow as-7.2 \rightarrow as-8.2$. As one can see, in this case, more backtracking steps are necessary to traverse the whole BI analysis graph.

We give a third example of a sequential analysis trace of the BI analysis graph of Figure 6.1. In this case, we assume that a business analyst decides in which temporal order the navigation steps are performed and the queries of the analysis situations are executed. Suppose, she or he is interested first to analyse urban and, afterwards, rural districts of a province: $as-1 \rightarrow as-2 \rightarrow as-3 \rightarrow as-4.1 \rightarrow as-7.1 \rightarrow as-8.1 \hookrightarrow as-3 \rightarrow as-4.2 \rightarrow as-7.2 \rightarrow as-8.2 \hookrightarrow as-4.1 \rightarrow as-5 \rightarrow as-6$.²⁴ In this case, user interaction is required to define the temporal order. One also can think of that the instantiation process of the user also represents the query execution and, finally, the generation of the sequential analysis trace.

Note that a BI analysis graph schema is used to instantiate one or more BI analysis graphs. In most cases, this is done by human interactions (for instance by business analysts). Partially, instantiation processes can be performed automatically, for instance, in the case of composite analysis situation

²⁴Both sequences $as-4.1 \rightarrow as-7.1 \rightarrow as-8.1$ and $as-4.2 \rightarrow as-7.2 \rightarrow as-8.2$ represent the analysis of urban districts whereas sequence $as-4.1 \rightarrow as-5 \rightarrow as-6$ represents the analysis of rural districts.

schemas. The specification of BI analysis graph schemas and the instantiation of BI analysis graphs represents more or less creative work of persons. On the other side, parallel and sequential analysis traces are only representations of BI analysis graphs that can be automatically created. Such traces represent execution models of the queries of the included analysis situations. Whereas each BI analysis graph induces a unique parallel analysis trace, for each BI analysis graph, several sequential analysis traces can be derived depending on the temporal order the queries of a BI analysis graph are executed. On the other side, if one decides for a specific traversal algorithm (for example, depth-first traversal), a sequential analysis trace also can be generated automatically in a unique way.

6.7 Discussion

BI analysis graphs and BI analysis graph schemas were defined in this chapter. Both constructs represent the main part of APMN4BI. The modeling of analysis processes is based on BI analysis graph schemas. The clear distinction between schema and instance level represents an essential aspect of APMN4BI. Modeling can be seen as a proactive and creative process that is performed by human beings (for instance, by business analysts). A business analyst must decide how general or specific a BI analysis graph schema has to be or which alternative analysis branches must be taken into account. She or he looks for appropriate analysis situation schemas and navigation step schemas, and she or he uses unbound variables and navigation guards to model meaningful and useful analysis processes at schema level. In this sense, a modeler predetermines what is allowed and meaningful to analyze. The purpose (depending on the user group of a model) and other restrictions (for example, restrictions concerning data privacy) have to be regarded in the elaboration of an APMN4BI model (BI analysis graph schema).

The instantiation of BI analysis graph schemas represents the use of analysis process models. Also for instantiation, some creativity²⁵ is required

²⁵Compared with the elaboration of BI analysis graph schemas, the instantiation of BI analysis graph schemas (the generation of BI analysis graphs) requires less creativity.

depending on the degree of freedom (mainly determined by unbound variables at schema level). A business analyst has to decide which parts of a BI analysis graph schema are interesting for the current analysis. Free variables have to be bound to appropriate values. Finally, the business analyst has to decide about the temporal order of instantiations (for instance, in accordance with priorities). The result of such an instantiation process is represented by a final BI analysis graph which can also serve as a kind of documentation of a specific analysis process.

Analysis situations of a BI analysis graph represent database queries that can be executed immediately after instantiation of an analysis situation schema or they can be executed later, when a BI analysis graph is traversed. Thus, a BI analysis graph can also be considered as an execution plan for database queries. Whereas a parallel analysis trace represents the tree structure of a BI analysis graph and reveals parallel execution paths, a sequential analysis trace defines a linear temporal order that specifies for each analysis situation a unique logical point in time when the database query of the analysis situation has to be executed. There are several possibilities to deduce such linear temporal orders. There may be algorithmic approaches as, for example, depth-first traversals, or the temporal order is specified by human beings, for example, in accordance with the temporal order deduced from the instantiation process.

This chapter also described possibilities to structure BI analysis graph schemas and BI analysis graphs. Subgraphs represent BI analysis graph schemas and BI analysis graphs that are embedded in other BI analysis graph schemas and BI analysis graphs. This can be done in an hierarchical manner. Thus, BI analysis graph schemas as well as BI analysis graphs can be recursively decomposed such that the general principle for managing complexity (“divide and conquer”) can also be applied for APMN4BI.

A specific kind of decomposition into subgraphs represents the specification of composite analysis situation schemas and composite analysis situations. On the one side, composite analysis situation schemas and composite analysis situations are subgraphs (BI analysis graph schemas or BI analysis graphs) per definition, on the other side, they can be considered conceptually

ally as analysis situation schemas and analysis situations, respectively. But composite analysis situation schemas and composite analysis situations additionally fulfill another purpose. A composite analysis situation schema can be instantiated as a whole in one go. Hence, it provides a possibility for a partial automation of the instantiation process meaning that all analysis situation schemas included in a composite analysis situation schema can be instantiated automatically in one go. A user only has to initiate the instantiation of the root analysis situation schema of the composite analysis situation schema. The instantiation of the remaining analysis situation schemas (of the composite analysis situation schema) can be automatically done without further user interaction.

In APMN4BI, there exists a precise distinction between schema and instance level. APMN4BI models are represented by BI analysis graph schemas that use constructs of the schema level, whereas BI analysis graphs are instances of BI analysis graph schemas that represent a specific analysis process that can be performed. Approaches toward this distinction already have been made in [90] and [91]. BI analysis graph templates in [90], and generic analysis situations and generic navigation steps in [91] can be considered as constructs at schema level.

In [91], BI analysis graphs were considered as a frame-oriented approach [30]. Generic and individual analysis situations co-exist in one BI analysis graph as a result of an ongoing development that reflects the nature of BI analysis processes as an evolutionary development process. Design and use phases of BI analysis graphs alternate.

In contrast to [91], the APMN4BI approach does not yet merge constructs of schema and instance level. APMN4BI models are represented by BI analysis graph schemas, and schema constructs can be more or less specific depending on the usage of a higher or lower number of free variables. Hence, APMN4BI propagates a proactive modeling approach. First models are defined proactively and afterwards they are used. This does not necessarily contradict to an exploratory and iterative process in data analysis. The development of APMN4BI models can and should be still performed as an evolutionary process. A business analyst specifies an initial model,

uses it for analysis, obtains new insights, extends and adapts it, etc. During this evolutionary process also the underlying eDFM may be adapted and extended.

Chapter 7

Evaluation

Contents

7.1	Influence of Evaluation on the Design	365
7.2	Case Studies	376
7.2.1	Brief Descriptions of the Case Studies	377
7.2.1.1	Case Study 1: LEICON	378
7.2.1.2	Case Study 2: KOTI Kobra	380
7.2.1.3	Case Study 3: PVA	382
7.2.2	Management Summaries of the Case Studies . . .	384
7.2.2.1	Case Study 1: LEICON	384
7.2.2.2	Case Study 2: KOTI Kobra	387
7.2.2.3	Case Study 3: PVA	389
7.2.3	Consolidated Evaluation	391
7.2.3.1	Non-comparative Analysis Situations and eDFM's	391
7.2.3.2	Comparative Analysis Situations	396
7.2.3.3	Navigation Steps and Navigation Oper- ators	398
7.2.3.4	Comparative Navigation Operators . . .	401
7.2.3.5	Derived Cubes	403

7.2.3.6	Extensions to Schema Level	404
7.2.3.7	Organization of BI Analysis Graphs	407
7.3	Evaluation of Claimed Design Criteria	409
7.3.1	Criteria 1: Domain Specific Conceptual Modeling Language	409
7.3.2	Criteria 2: Coherent Language Design	413
7.3.3	Criteria 3: Completeness of Model Constructs	415
7.3.4	Criteria 4: Mapping to SQL	418
7.3.5	Criteria 5: Early consistency checking	420
7.3.6	Criteria 6: Reasonable Decisions Regarding Trade- offs	421

This chapter presents the evaluation of the design of APMN4BI as presented in this thesis. The evaluation itself had an impact on the design of APMN4BI. Evaluation and development of APMN4BI can be considered as an iterative research process. Thus, we start with an introductory section that presents the influence of the evaluation on the design of APMN4BI. Intermediate results, gained insights, deficiencies, and improvements that emerged during this iterative research and evaluation process, and that led to the final version of APMN4BI are discussed. Section 7.2 comprises three final case studies. The environments of these case studies were already introduced in Section 1.3 of Chapter 1 and are taken from Austrian public health insurance organizations, from brush manufacturing, and from Austria’s public pension insurance organization. Brief descriptions, management summaries, and consolidated evaluation concerning the final case studies can be found in separate subsections. Finally, in Section 7.3, the design criteria presented in 1.4.2 are evaluated.

7.1 Influence of Evaluation on the Design of APMN4BI

The development of APMN4BI was based on an iterative research and integrated evaluation process. Concepts of APMN4BI were elaborate on the basis of real use cases, intermediate results were evaluate in real environments, and gained insights and improvements were incorporated in further development that can be considered as a subsequent iteration of the research and evaluation process.

In the context of the research project *semCockpit*¹, use cases and field studies were provided by an Austrian and a German public health insurance organization: *Oberösterreichische Gebietskrankenkasse (OÖGKK)*² and *Deutsche Angestellten Krankenkasse (DAK)*. Although it was not the aim of *semCockpit* to develop an analysis process modeling notation for business intelligence, the importance to document and model the navigation from one to another analysis situation as a part of an overall analysis process was recognized. First approaches of BI analysis graphs were elaborated and evaluated on the bases of use cases provided by *OÖGKK*³ and *DAK*.

The first approach about BI analysis graphs was published 2012 in [90] in the context of *semCockpit*. BI analysis graphs were already considered as artifacts for conceptual modeling that leads to multi-dimensional navigation modeling, a conceptual approach suitable for OLAP structures. Concepts as analysis situations and navigation operations already exist, navigation steps are denoted as analysis steps, and analysis situations are based on multi-dimensional content models (comparable with dimensional fact models). BI analysis graph templates can be considered as first rudiments towards BI analysis graph schemas. The definition of an analysis situation is based on a point-centric view (a view generally used in *semCockpit*) that comprises a multi-dimensional point, a measure, a qualification (predicates), and a group-

¹started in March 2011 and finished in February 2014

²After a reorganization of Austria's public health insurance organizations, *OÖGKK* and other public health insurance organizations were consolidated to one public health insurance organization that, since 2020, is called *Österreichische Gesundheitskasse (ÖGK)*.

³Especially, use cases were provided by the project/program *LEICON*.

ing granularity. A navigation operation represents the difference between two analysis situations which already led to the phrase “navigation is knowledge”. For evaluation, simple use cases of the LEICON project (OÖGKK), especially analysis of drug prescriptions, were used.

This first approach of multi-dimensional navigation modeling was refined as a part of the overall semCockpit approach that focused on ontology-driven business intelligence for comparative data analysis. A comprehensive presentation of all semCockpit concepts is included in [91]. This refined approach comprises dimensional fact models where dimension levels are based on entity classes and where multi-dimensional ontology (MDO) concepts are included. The notion of analysis situations was already extended to the distinction between non-comparative and comparative analysis situations. By the introduction of comparative analysis situations also the notion of scores was specified. Comparative analysis situations represent an important concept necessary to realize comparative data analysis which was among others a main goal of semCockpit. On the other side, the semCockpit approach is an ontology-driven one (another main goal of semCockpit) and, thus, a point-centric view was still retained. Composite analysis situations are also already presented in [91] as another concept that has been transferred to APMN4BI. Navigation steps (a notion introduced in [91]) are defined by navigation operators. The set of navigation operators was extended, especially also to take into account comparative analysis situations. Although, in this stage of the semCockpit approach, we distinguished between generic analysis situations and individual analysis situations and between generic navigation steps and individual navigation steps, the notion of a BI analysis graph did not yet make a clear separation between schema and instance level as it is done in APMN4BI. Design steps and use steps were presented as more or less alternating steps, and design is additionally supported by specialization in the sense of inheritance. This is different to the proactive modeling approach as it is proposed in APMN4BI where a BI analysis graph schema is modeled and, afterwards, this model can be used to instantiate BI analysis graphs. The elaboration at this stage was mainly encouraged by LEICON use cases, especially by analysis of patients having diabetes mellitus of type 2 (DM2

patients).

One insight gained from field studies of the semCockpit project (carried out at OÖGKK and DAK) was that there is a need for intelligent guidance in the area of business intelligence. Discussions and the everyday work with, especially, long-term customers showed that the theoretical view about BI analysis graphs as it was developed in semCockpit has to be adapted to a “more practical” one that also allows an easier understanding of and an easier usage by business analysts and subject matter experts that are in the role of a BI user. Thus, after the end of the semCockpit project, the elaboration of BI analysis graphs towards a “more practical approach” was continued. The two Austrian public health insurance organizations, Oberösterreichische Gebietskrankenkasse (OÖGKK) and Niederösterreichische Gebietskrankenkasse (NÖGKK)⁴, provided further input and discussions in the context of LEICON.⁵ Additional input and discussions from another industry (brush manufacturing) was provided by the long-term customer KOTI Kobra. Since 2016 the Austrian pension insurance organization Pensionsversicherungsanstalt (PVA) was a new customer of solvistas GmbH that also provided further input.

One adaptation concerns the change from the point-centric view of semCockpit towards a set-oriented view. The idea of multi-dimensional points to which measures and MDO concepts are applied follows an ontological approach that is unusual for users in the area of business intelligence. For such a user group, a set-oriented view is more common. It corresponds more to the thinking that an analysis situation represents a query (especially a database query in SQL) that returns a query result set after execution. A second general change addresses the vague view of mixed consideration of generic and individual elements (generic analysis situations and generic navigation steps, and individual analysis situations and individual navigation steps). In [92], a proactive modeling was proposed that respects a clear distinction between

⁴Note that since 2020, OÖGKK and also NÖGKK are integrated organizationally as regional sub-organizations in the public health insurance organization Österreichische Gesundheitskasse (ÖGK).

⁵OÖGKK is responsible for the technical development and NÖGKK has the business- and application-specific lead of LEICON.

BI analysis graph schemas and BI analysis graphs. Whereas a BI analysis graph schema represents a model of a class of analysis processes that is developed proactively at schema level, an instance of such a BI analysis graph schema represents an individual BI analysis graph that can be considered as a specific analysis process which is executed. The alternating design and use steps of semCockpit (compare [91]) can be still considered as an iterative development process of a BI analysis graph schema⁶ but, conceptually, a clear distinction between the result of such a development process (a BI analysis graph schema) and a specific usage of this result (a BI analysis graph) has to be made. In [92], use cases are taken from brush manufacturer KOTI Kobra (another industry apart from public health insurance organizations).⁷ This has also enabled to evaluate the APMN4BI approach to other application areas of business intelligence.

Although, up to this point, main concepts of BI analysis graphs (non-comparative and comparative analysis situations, navigation operators and navigation steps, and composite analysis situations) were already established, the idea of an analysis process modeling notation for business intelligence was elaborated and refined after the semCockpit project and on the basis of the presentation in [92]. Especially, the graphical representation of an enriched dimensional fact model, of analysis situations and navigation operators are important to use APMN4BI as a modeling notation. At this time also the notion of APMN4BI was born. This graphical notion was developed and refined, and an exact formalization and exact definitions were elaborated. In this phase of the thesis, it was essential that also feedback from customers (especially from OÖGKK and NÖGKK concerning LEICON, from KOTI Kobra, and from PVA) about usability and understandability of APMN4BI could be collected. Furthermore, valuable input could be also obtained from students of practical courses given between 2014 and 2017 where parts of a modeling tool for APMN4BI were implemented and evaluated. Pictograms were defined and evaluated to associate visually the meaning and usage of elements

⁶Such an iterative development process can be interpreted in the following way: create an initial BI analysis graph schema, use it, perhaps adapt or extend it, use it again, make further adaptations or extensions, etc.

⁷Use cases from KOTI Kobra can also be found in [113].

of APMN4BI. For instance, each navigation operator has obtained a specific pictogram that symbolizes its effect, non-comparative and comparative navigation operators are visualized differently, non-comparative navigation operators are also graphically included in comparative navigation operators; but also the various types of analysis situations and the various components of analysis situations have obtained specific symbols. Moreover, a consistent use of pictograms was introduced. Same or similar symbols for elements of an eDFM are used for components in analysis situations. For example, the symbol for a dimension level in an eDFM represents the basis for the symbols of a dice level and granularity level including minimal appropriate graphical extensions, or the pictogram for a dimensional predicate is the same as for a slice condition. Furthermore, such symbols are also re-used as parts of pictograms for corresponding navigation operators. Also the distinction between schema and instance level was included in the graphical representation (double-edged versus single-edged borders for analysis situations). The full and condensed graphical representation of analysis situations were complemented by a lean notion that contains all information of the full notation but which is more compact and, thus, more pleasant to read. Furthermore, the concept of navigation guards was introduced to provide additional means of control. As a consequence, the need for a “discriminate operator” to specialize analysis situation schemas (in the sense of inheritance) could be avoided because such case differentiation can also be realized by navigation guards. In practice, we made the experience that navigation guards are more understandable for BI users than the concept of inheritance (specialization and generalization).

Finally, in 2018 three case studies were performed to evaluate the APMN4BI approach: one case study in the context of LEICON (especially including public health insurance organizations NÖGKK and OÖGKK), another one in the context of manufacturing at KOTI Kobra, and a final one in the context of public pension insurance at PVA. In each case study, two different analysis processes were modeled and evaluated. The insights gained from this evaluation were documented as internal reports. Some insights from this final evaluation were still incorporated as improvements into APMN4BI.

Other aspects were left open and can be considered as impulses for further development of APMN4BI.

One improvement based on the final case studies concerns the graphical notation of actual parameters, the naming of navigation operators, and “decorations” of actual parameter. In all three case studies, we already used the same or similar graphical symbol for elements in an eDFM that are used as elements in an analysis situation. That means, for instance, that the pictogram used for a dimension schema in an eDFM is also used for the dimension qualification in an analysis situation (schema), or the pictogram of a dimension level in an eDFM is used as a basis for the pictogram of a dice level, a dice node, and a granularity level in an analysis situation (schema). Note, the symbol for a dimension level in an eDFM is the same as for a dice level, the symbol for a dice node additionally includes a “black point”, and the symbol for a granularity level additionally includes “three small circles”. On the other side, for actual parameters, we used no symbols at all. Instead of pictograms, we used “abbreviations” together with the name of the actual parameter. For example, we wrote as actual parameters such like **Dim** = *Time* for dimension schema *Time*, **DL** = *year* for dice level *year*, **DN** = 2016 for dice node 2016, or **GL** = *insDistrict* for granularity level *insDistrict*. This notation for actual parameters was still used in the KOTI Kobra use case, and in the first example of analysis processes of the LEICON case study. In the second analysis process of the LEICON case study and in the PVA case study we already used pictograms instead of such abbreviations. This change to the usage of pictograms in the context of actual parameter was a consequence of the feedback gained from the final case studies.

Another change refers to navigation operators that narrow, broaden, or refocus slice conditions, base measure conditions, filter conditions for aggregate measures, and score filters. In the version of APMN4BI that was used in the context of the three final case studies, these navigation operators were overloaded⁸ and actual parameters were decorated with additional sym-

⁸Overloading of navigation operators is used in the same sense as function or method overloading in programming languages, i.e., for different functions or methods, the same function/method name is used and on the basis of the number and types of parameters, the final decision can be made which function/method is meant.

bols that actually completed the operator's definition. For instance, name `narrowSliceCond` was used for two navigation operators that narrow slice conditions; one operator narrows slice conditions by adding additional dimensional predicates (given as actual parameters prefixed by symbol `+`), the other one narrows slice condition by exchanging a dimensional predicate of the slice condition by another one that implies the old one (given as actual parameters connected by infix-symbol `->`). During the execution of the final case studies, we gained the insight that the overloading of navigation operators together with decorating actual parameters by specific symbols is more confusing than clarifying for BI users. Thus, we changed the operators' names to different names that include this specific symbols in their names: `narrowSliceCond+` and `narrowSliceCond->`. The decorating symbol `+` as a prefix for actual parameters is omitted but the infix symbol `->` is still used because it symbolizes the change from one to another dimensional predicate in a comprehensible manner. A similar change to the operator name `broadenSliceCond-` was accomplished. Furthermore, analogous name changes were performed for similar navigation operators that change base measure conditions, filter conditions for aggregate measures, and score filters. Moreover, now we also use the symbol `->` for navigation operator `refocusMeasure->` to express that one measure is exchanged by another one.⁹

Each case study also included comparative analysis situation schemas and comparative navigation step schemas. We realized that join conditions were the only items that occurred as inline expressions¹⁰ for properties in comparative analysis situation schemas as well as for actual parameters of comparative navigation operators. Thus in the final design of APMN4BI, join conditions were also incorporated in the definition of an eDFM which means that join conditions obtain a name which is defined in the underlying eDFM and that can be used as a property of an analysis situation schema as well as an actual parameter of a comparative navigation operator. As a consequence, we obtained a consistent approach in the sense that all items

⁹In general, for detailed syntax in the operator naming, see Chapter 4.

¹⁰Similar to programming languages, an inline expression represents an expression that is not referenced by a name but that is used directly.

in an analysis situation schema refer to defined constructs in an eDFM—consistency in this sense also was a general demand taken from the case studies.

On the other side, we generally do not use inline definitions for properties of analysis situation schemas and for actual parameters in navigation step schemas. Thus, all items that are used as properties of analysis situation schemas and actual parameters have to be defined as named objects in the underlying eDFM. We discussed such inline definitions during the case studies. Although sometimes it would be convenient to use inline expressions as ad-hoc items for properties of analysis situation schemas and for actual parameters, there was no convincing consensus for allowing them in general. Inline expressions would increase the complexity of the formalization of APMN4BI and its graphical representations, and, moreover, an advancement of defining reusable items in an eDFM would be undermined. Hence, we refrained from incorporating such inline definitions in APMN4BI.

Revisiting join conditions as actual parameters of comparative navigation step schemas, there was an attempt in the case studies to use equi-joins (between the same dimension schemas of the context of interest and the context of comparison) as “default joins” in the sense that one can omit these standard joins as actual parameters at least in the graphical representation of navigation step schemas. Nevertheless, we refrained from the use of such standard joins because an additional issue would arise as one has to distinguish equi-joins as default joins from the cartesian product (meaning there is no join condition) that also could occur in comparative analysis situation schemas (for example to compare every item of a dimension in the context of interest with all items of the same dimension in the context of comparison).

In a former version of APMN4BI that also was the basis for the final case studies, unbound variables in analysis situation schemas were only permitted for the set of base measure condition, for the set of filter conditions, for a dice level, for a dice node, for a set of slice condition, for a granularity level, and, in the case of a comparative analysis situation schema, for the set of score filter conditions. For most analysis processes in these case studies, using variables for these properties was sufficient except for one analysis process

in the LEICON case study where variables for the set of measures were necessary. As a consequence, we generally revisited the APMN4BI formalism and extended the definition of analysis situation schemas such that for all properties it is allowed to use variables except for the cube schema and for dimension schemas. For this change, it was also necessary to extend the definition in such a way that both the cube schema and a cube instance have to be properties of a non-comparative analysis situation schema. Now it is possible that even the cube instance can be a variable (but not the cube schema).

The analysis processes of the case studies of KOTI Kobra and LEICON included navigation guards. Most navigation guards in the KOTI Kobra case study concerned such ones that examine the result set of source analysis situations. Some navigation guards in both case studies checked properties of the source analysis situation. In these cases, there was the attempt to refer to the properties of the source analysis situation schema by named variables, i.e., variables used as properties in source analysis situation schemas were also used in the boolean expression of navigation guards. Especially in the case of condensed graphical representation of analysis situation schemas, this became a problem because the variables of a source analysis situation schema depicted in a condensed graphical representation were not visible. Thus in the final version of APMN4BI, for navigation guards, a rigorous use of operators¹¹ is preferred that return the values of the properties of analysis situations.¹² Furthermore, the definitions of navigation step schemas and navigation steps were revised such that navigation guards are inherent constituents of navigation step schemas and navigation steps.

In the final case studies, it could be realized that there is a need for structuring BI analysis graph schemas and BI analysis graphs. The concept of subgraphs provided adequate means to preserve an overview of all modeled analysis processes and subprocesses. This was the motivation to introduce subgraphs for structuring analysis processes at both schema and instance

¹¹compare Section 5.2.2 of Chapter 5

¹²One can compare such operators analogously to “getter-methods” in object-oriented programming languages.

level. Thus, also an appropriate graphical formalism was introduced that distinguishes graphical representations of subgraphs at schema level as well as at instance level.¹³

Although composite analysis situations were introduced in an early stage of APMN4BI, further refined reflections were made on the basis of final case studies. Composite analysis situation schemas were used in the KOTI Kobra and in the LEICON case study. In some cases, the primary intention for using composite analysis situation schemas was to structure the overall analysis process. For this goal, structuring into subgraphs would have been already satisfactory—the specification as composite analysis situation schemas was not necessarily indispensable in these cases. On the other side, composite analysis situations schemas also occurred where the instantiation as a whole and in one step was a meaningful demand. After these case studies, composite analysis situation schemas and composite analysis situations were formally defined as subgraphs with further specific properties. Furthermore, also some graphical improvements were made. The root analysis situation schema and the root analysis situation (at instance level) are decorated now by an additional small square. By this decoration the root can be identified immediately, also if there are no incoming navigation step schemas which is allowed in the final version of APMN4BI. Moreover, now input and output navigation step schemas (and input and output navigation steps) also obtain a label to increase understandability.

Finally, the KOTI Kobra case study caused another adaptation of APMN4BI with respect to the repertoire of navigation operators. This adaptation concerned the move operators that can be used to iterate over a set of dimension nodes with respect to one parent node. The originally existing navigation operators `moveToFirstNode`, `moveToLastNode`, `moveToNextNode`, and `moveToPrevNode` only allow to iterate over subnodes of a dimension node that belong to the direct superlevel of those subnodes. In the KOTI Kobra case study, it was also necessary to iterate over subnodes that do not belong to the direct sublevel but to a finer grained sublevel. Thus, the navigation operators were extended to overloaded versions of operators `moveToFirstNode`,

¹³double-edged versus single-edged boarders

`moveToLastNode`, `moveToNextNode`, and `moveToPrevNode` which additionally take a level that allows to iterate over nodes that belong to a dimension level with a difference greater than one to the dimension level of the parent node (see Chapter 4).

Although many results and insights gained from the evaluation influenced the final design of APMN4BI, there are some aspects that were left open in this thesis. The goal of this thesis was to demonstrate the main ideas of the APMN4BI approach. Thus, a middle course was steered between focusing on the main ideas of APMN4BI keeping concepts as simple as possible, and incorporating further constructs that increase complexity in the formalism and that obscure the essential approach of APMN4BI. In the remaining paragraphs of this section, further aspects are listed that arose as a result from the three final case studies but that were not incorporated in APMN4BI.

Some suggestions gained from the case studies concerned the definition and use of dimension schemas of an eDFM. The KOTI Kobra and LEICON case study gave rise to use dimension schemas as namespaces for dimension levels and descriptive attributes. In APMN4BI, unique global names for dimension levels, dimensional operators, and descriptive attributes are required. A similar and more general suggestion came up from the PVA case study where dimension roles would have been a useful construct. Quite similar to such suggestions, all three case studies showed that the introduction of entity types as a basis for dimension levels (and associated descriptive attributes) would facilitate reusability and consistency with respect to unique definitions of dimension levels. Dimension hierarchies represented another point of discussion, especially in the PVA case study. In APMN4BI, a dimension schema comprises exactly one dimension hierarchy. Thus, the demand of more than one dimension hierarchy in one dimension schema has to be solved in APMN4BI by defining more dimension schemas, i.e., one dimension schema for each hierarchy.

Furthermore, the KOTI Kobra and the LEICON case study demonstrated that the definition and use of multi-ary predicates for slice conditions, base measure conditions, filter conditions, and score filter conditions would increase usability of APMN4BI. By additional parameters, such predicates can

be used more universal. The PVA case study also presented examples where, in combination with the `drillAcrossToCube`-operator, the result set of a source analysis situation is used for selecting dimension nodes in the target analysis situation which represents another type of predicate not available in APMN4BI.

Sequences of navigation operators represent another construct that is not defined explicitly in APMN4BI. All three final case studies comprised examples which demonstrated that there are more than one navigation operators (a sequence of navigation operators) from an interesting source analysis situation to an interesting target analysis situation. This means that intermediate analysis situations between the source and the target are not of interest. Sometimes such intermediate analysis situations are even meaningless. In this sense, such operator sequences can also be recreated in APMN4BI by ignoring intermediate analysis situations that are formally and visually present. Nevertheless, a formal and graphical representation of such sequences of navigation operators as a separate construct would yield a better formal and visual appearance.

In the context of navigation guards, we already presented operators that return values of a source analysis situation (comparable with getter-methods in object-oriented programming languages). The use of such operators could be extended to actual parameters of navigation operators. The advantage of such getter-methods compared with the use of named variables was already mentioned previously in the context of navigation guards. Furthermore, in the final version of APMN4BI, we did not discuss and formalize at all the use of expressions as actual parameters. These considerations concerning operators as getter-methods of analysis situations and expressions as actual parameters occurred in all of the three final case studies.

7.2 Case Studies

In this section, three final case studies are presented that were used as a final evaluation and that contributes to a final revision of APMN4BI. These case studies were taken from the areas of public health insurance, of brush

manufacturing, and from public pension insurance. Because of the lack of an appropriate modeling tool, APMN4BI models were sketched with pencil and paper, and drawn by the visualization tool Microsoft Visio. These visualization provided a sufficient basis for gainful discussions and for useful elaboration and evaluation cycles. Although several prototype implementations in university courses provided additional insights for APMN4BI, they did not accomplish workable tools for end-users.

Subsection 7.2.1 presents a brief description of each case study. The results of these case studies were documented as internal reports (written in German language) which comprise management summaries that can be found in Subsection 7.2.2 as versions translated into English. In Subsection 7.2.3, a consolidated evaluation over all three case studies is presented.

7.2.1 Brief Descriptions of the Case Studies

The following subsections give descriptions of the final case studies performed for evaluation of APMN4BI. These case studies concern analysis processes in the area of Austrian public health insurance organizations, in the area of brush manufacturing, and in the area of Austria's public pension insurance organisation. They were performed in year 2018 and gained insights were used for further improvement and refinement of APMN4BI. The first case studies was performed in the context of working program LEICON¹⁴ (also considered as a “data warehouse product”) which was in the responsibility of the Austrian public health insurance organizations Niederösterreichische Gebietskrankenkasse (NÖGKK), Oberösterreichische Gebietskrankenkasse (OÖGKK), and Versicherungsanstalt für Eisenbahnen und Bergbau (VAEB).¹⁵ Brush manufacturer KOTI Kobra was the partner in the second case study and the third case study was performed at the Austrian public

¹⁴abbreviation for “LEIstungsCONtrolling in der Sozialversicherung”

¹⁵Since 2020 NÖGKK and OÖGKK are integrated as regional sub-organizations in the public health insurance organization Österreichische Gesundheitskasse (ÖGK), and VAEB becomes a sub-organization of the reorganized public health insurance organization Versicherungsanstalt öffentlich Bediensteter, Eisenbahnen und Bergbau (BVAEB). Now LEICON is serviced by the successor organizations ÖGK (main project partner) and BVAEB.

pension insurance organization Pensionsversicherungsanstalt (PVA).

Each case study demonstrates two different analysis processes that are presented textually and modelled by APMN4BI. The analysis processes and the insights are summarized as internal reports that are written in German language and which are reported in an external document as appendices to this thesis [87, 88, 89]. APMN4BI models created in these case studies were drawn by Microsoft Visio. Note, that the diagrams in the internal reports may differ from the notation introduced in this thesis. The reason for this is that the final notion was also adapted on the basis of the insights gained from the final case studies.

7.2.1.1 Case Study 1: LEICON

The first case study concerns the DWH product LEICON as already introduced in Section 1.3.1. It was performed from March to July 2018. In LEICON, data of thirteen Austrian public health insurance organizations¹⁶ (mainly clearing data) is collected for disease pattern related analysis. Measures are calculated per year, for instance to analyze patients having diabetes mellitus type 2 (DM2). The case study is presented in detail in [87] (Appendix A) in German language and comprises two real analysis processes. In this section, we only give a short description of the LEICON case study.

Data used for LEICON itself can be considered as a data warehouse that uses data from thirteen other data warehouses named FOKO (see Section 1.3.1) where each FOKO data warehouse belongs to one Austrian public health insurance organization. This data has to be imported, consolidated, and aggregated to obtain the required measures.

The first analysis process can be considered as a quality assurance (QA) process. It is necessary to ensure that the source data (FOKO) is sufficiently reliable. The clearing data provided by FOKO comprises four types of services: drug prescriptions, ambulant treatments, hospitalizations, and ambu-

¹⁶Note, after reorganization in 2020, the thirteen Austrian public health insurance organizations were integrated in three bigger public health insurance organizations: Österreichische Gesundheitskasse (ÖGK), Versicherungsanstalt öffentlich Bediensteter, Eisenbahnen und Bergbau (BVAEB), and Sozialversicherungsanstalt der Selbständigen (SVS).

lance services. Different services can be analyzed with respect to different dimensions. For example, drug prescriptions can be analysed with respect to the ATC hierarchy and hospitalizations with respect to ICD10 coding (see also Section 1.3.1). The internal report of this case study comprises an eDFM about drug prescriptions where the drug dimension comprises the ATC hierarchy and dimensional predicates for DM2 specific medication (e.g., oral antidiabetic drugs, oral antidiabetic drugs for combination therapy, or psychotropic drugs). Measures and measure filters assessing deviations are defined over drug costs and the amount of drug prescriptions.

The analysis process (QS process) refers to an entered year, compares measures with the previous years, and applies filters to detect significant deviations between the actual and previous years. The whole analysis process is divided into subprocesses that separately analyze drug prescriptions, ambulant treatments, hospitalizations, and ambulance services. Moreover, comparison is refined with respect to several criteria, for example, per quarter, with respect to specific types of drugs (for instance, for oral antidiabetic drugs), or, for instance, with respect to diabetes relevant hospital diagnoses. This analysis process was elaborated and modeled in APMN4BI in several iterations to obtain general subprocesses, among others by introducing more variables, for instance, for slice conditions restricting to DM2 specific medication.

In the second analysis process of this case study, measures about DM2 patients of a year that are compared with previous years to detect striking differences. LEICON calculates thousands of measures per year. In the analysis process of this case study, we restricted to 25 measures that can be grouped into prevalence measures (e.g., overall prevalence of DM2 patients, prevalence of DM2 patients obtaining oral anti diabetic drugs, or high-risk patient for DM2), cost measures (e.g., total costs per patient, drug prescription costs per patient, or ambulant treatment costs per patient), supplying measures (e.g., amount of patients obtaining HbA1c screening, amount of patients visiting oculists, or, amount of patients obtaining HDL-/LDL-cholesterol screening), and measures for assessing process quality (e.g., amount of patients with visiting general practitioners in at least three quarters, amount of patients with

“first DM2 therapy”, or amount of patients with at most one HbA1c screening). These measures can be analyzed, for instance, with respect to DMP-versus Non-DMP-patients, with respect to insurers, with respect to regional segmentation, and with respect to time. The measures are distributed over two cubes modeled by two eDFM’s. Concerning the second analysis process, in this case study, two subprocesses are depicted in APMN4BI, one that analyzes significant deviations of prevalence and cost measures, and one that analyzes supply and process quality measures.

7.2.1.2 Case Study 2: KOTI Kobra

In the second case study, another subject area was inspected. KOTI Kobra is a company that produces brushes as described in Section 1.3.2.¹⁷ One real analysis process concerns the procurement process of material and the other one is used for a monthly profit analysis. This case study was conducted from March to May 2018. An internal report about it can be found in [88] (Appendix B). Both analysis processes can be carried out on the basis of IBM Cognos reports that were implemented and that are used by KOTI Kobra. Although model elements of APMN4BI are not available in these reports, these elements were imitated as well as possible. For example, the use of dimensional predicates for narrowing slice conditions was simulated by entering and applying appropriate search criteria.

The goal of the first analysis process concerns effective planning for material procurement. As an example, one can consider demand-based procurement of Chinese nylon used for production of a certain type of brushes. On the one hand, the material price represents an important calculation factor, on the other side, it is important to ensure a just in time supply and a sufficient quality of the ordered material. This analysis process and the related reports are based on a cube that contains costs and quantities of material used for production in the past. The underlying eDFM comprises dimensions for customers, final products, production material, production order, and time (date of the creation of a production order). These dimen-

¹⁷The Austrian company KOTI Kobra is a subsidiary of the European company group KOTI with headquarters in Netherlands.

sions comprise several dimension levels, for instance, several levels to classify costumers to various industries or several levels to classify final products to various product types. The eDFM is also enriched by an aggregate measure for summing up quantities, and by several measure filters and dimensional predicates. Measure filters and dimensional predicates are elements that cannot be found in this form in the reports of KOTI Kobra. As already mentioned these constructs are simulated by appropriate input for search criteria or simply by manual user actions (for instance, manual selection of specific rows in a report).

In this first analysis process, a specialist of KOTI Kobra enters a year in the past for analysis and narrows the material for bristles stepwise to nylon and nylon from China. Only significant material consumption is analyzed in detail. Also the regularity over a year is regarded by additionally performing analysis per quarter. Furthermore, the final products and costumers are analyzed to obtain information for what and for whom this material is used. This is important because in coordination with sale, the procurement specialist can conclude the material required.

The second demonstrated analysis process of KOTI Kobra comprises a monthly profit analysis that has to be reported monthly to the company group KOTI. For this analysis process another cube is used: a sales cube with base measures for quantity, revenue, and costs. There are dimensions that refer to customers, final products, and invoice date. The eDFM comprises aggregate measures for summing quantity, revenue, and costs, and for calculating the profit. There are also dimensional predicates, scores, and measure and score filters. Analogously to the material example, this analysis process can also be reproduced by using existing IBM Cognos reports.

In the second analysis process, a specialist of KOTI Kobra enters the latest completed month and retrieves the revenue of this month per product type. The revenue is analyzed at different product type levels. Afterwards, she or he compares the revenue with the same month of the previous year and looks at the relative deviation. In the following steps, only those product types that exhibit significant revenue are considered: The total revenue of the years and the revenue of all months of the year are compared. Finally,

these revenues are analyzed for all customers.

7.2.1.3 Case Study 3: PVA

The third case study was performed between June and September, 2018, in Austria's public pension insurance organization. One characteristic of this case study was that the real analysis examples themselves were in development. Cubes, reports, and analysis processes were developed parallel to the case study. A description of the setting for this case study can be found in Section 1.3.3 and the internal report about this case study is copied in [89] (Appendix C). Two real analysis processes were selected. In the first analysis process, planning of prospective rehabilitation institutions with focus on ambulant rehabilitation using an interactive map was considered. The second analysis process concerns a long-term evaluation of the effectiveness of rehabilitation.

The goal of the first analysis process is to analyze the care of Austria's population with rehabilitation institutions, especially with ambulant rehabilitation. Such an analysis has to take into account the geographical accessibility of rehabilitation institutions. The reachability was classified into three sectors: reachable within 15, 30, and 45 minutes of travel time. The visualization was supported by an interactive map (self-development of PVA) that also allows to plan potential rehabilitation institutions which also takes into account demographic development. This analysis and planning process also has to take care of other important constraints like indication categories (for instance, disorder of skeletal and locomotor system, cardiovascular disease, or psychiatric disorder) or rehabilitation types (distinction between ambulant and in-patient rehabilitation).

First a business analyst starts to visualize all rehabilitation institutions on the interactive Austria map. Afterwards, she or he moves down along dimension hierarchies to ambulant rehabilitation and to various indication categories. Finally, such a selection of rehabilitation institutions is taken to visualize the geographic area with respect to reachability (classified into 15, 30, and 45 minutes of travel time) and to compute the population served by these

institutions. Such an analysis process was modeled in APMN4BI whereby it was elaborated iteratively in an evolutionary approach (exploratory data analysis). This means, starting from specific analyses representing BI analysis graphs (for example, analyzing only rehabilitation institutions for disorders of skeletal and locomotor system), generalizations were developed stepwise (for instance, introducing a variable for indication category) that yield BI analysis graph schemas. These BI analysis graph schemas were instantiated, and, again, extended and improved by further experiments and insights.

For this analysis process, an eDFM was elaborated that comprises two cubes: a cube for rehabilitation institutions containing the capacity for rehabilitation and a cube for distances between municipalities (measured in travel time). Moreover, various dimensions and dimension levels were defined: dimensions for rehabilitation type, indication category, and various dimensions for geographical structuring. The population and the geographic coordinates of municipalities were provided by descriptive attributes. For classifying travel time, base measure conditions were specified that restrict to distance equal to or less than 15, 30, and 45 minutes of travel time.

In order to be able to objectively prove the effectiveness and sustainability of rehabilitation, the PVA is interested in demonstrating the effect of rehabilitation stays on occupational disability pensions. This view correlates with the PVA's aim of maintaining the ability of its customers (insurants) to work for as long as possible by preventive measures (by effective rehabilitation measures). The second analysis process in this case study refers to this goal. Again, this analysis process was a subject of a current BI development of the PVA.

Again, for this analysis process, a specific eDFM was defined. Two cubes are provided: a cube for rehabilitation stays and one for occupational disability pensions. Several dimensions are specified: indication category, insurants' age and sex, date for rehabilitation stays and the beginning of occupational disability pensions, and ICD10 diagnoses. There are also dimensional predicates, for instance, a predicate that restricts to insurants with relevant ages for analysis (age between 18 and 65 years) and scores that defines the calculation of absolute and relative difference of the number of insurants.

The analysis process starts with rehabilitation stays and computes the number of insurants (with relevant ages for analysis) having a rehabilitation in a year. Afterwards, the beginnings of occupational disability pensions after two years are considered by comparing insurants having rehabilitation and insurants without rehabilitation. In another analysis branch, one can additionally look at specific ICD10 diagnoses. Especially, it is interesting whether an insurant has the same diagnosis for obtaining a occupational disability pension compared with the diagnosis she or he has for rehabilitation. In both analysis branches, additional distinctions can be made, for instance, comparison of rehabilitation and occupational disability pensions with respect to sex.

7.2.2 Management Summaries of the Case Studies

The following subsections represent an English translation of the management summaries of the case studies that are contained in an external document as appendices to this thesis [87, 88, 89]. These management summaries are parts of the internal reports which were drawn up as a brief case study documentation. The internal reports were intended as a documentation for the “case study partners” (LEICON project members, KOTI Kobra employees, and PVDWH project members) and, thus, they were written in German, especially, because many subject specific German terms were used in these case studies. Hence, in this thesis, the internal reports were copied in German language in [87, 88, 89] and only the management summaries were translated into English and presented in the following subsections.

7.2.2.1 Case Study 1: LEICON

As an evaluation goal, the data warehouse standard product LEICON was used to check whether real analysis processes from the Austrian public health insurance sector can be modeled with APMN4BI and whether these models are useful.

LEICON exists since 2004 and deals with secondary data analyzes from the service areas of health insurance organizations. In particular, analyzes

are carried out in connection with disease patterns. LEICON has extensive data and key figures with respect to structure and quantity. Therefore, LEICON offers a great number of options for modeling real analysis processes in APMN4BI and evaluating them with regard to benefits. In the present case study, a small excerpt from the LEICON environment was used. The first example relates to an upstream quality assurance (QA) process, the second to the analysis of specific key figures that were determined in LEICON.

The part of the QA process modeled in APMN4BI can be viewed as a specification or documentation. An APMN4BI runtime environment could implement this QA process. Without such a runtime environment, the specification can serve at least as a template to expand the existing “QA machine”¹⁸ in order to be able to carry out a semi-automated QA. The five-year comparison in the case of an abnormality can be viewed as a “manual analysis” (analysis of a result report provided by the QA machine). The APMN4BI model represents a complete specification/documentation for both the automated and the manual part.

For some of the DM2 key figures, an analysis process was modeled in APMN4BI in order to discover conspicuous deviations and to carry out other interesting analysis actions (comparison of DMP and non-DMP patients; analysis at province and district levels). In addition, different analysis steps are required depending on the respective key figure (e.g., analysis per carrier for the prevalence and cost indicators; regional analysis and DMP / non-DMP analysis for the PQINB key figures¹⁹). The APMN4BI model serves as a specification / documentation of the process for key figure analysis, which is to be carried out annually. This documentation makes it easier for the same “accuracy and attention” to be applied—even in the case that this analysis has to be carried out by a “novice”. It should be pointed out again that with the large number of different key figures, an exact specification of analysis processes is all the more important and, therefore, the benefits of

¹⁸An Austrian public health insurance organization has implemented a “QA machine” (in German: “Prüfmaschine”) for data warehouses that examine data with respect to certain categories of data quality rules.

¹⁹PQINB is a German abbreviation for “Prozessqualität im niedergelassenen Bereich” (process quality in the ambulant area).

the APMN4BI models can be emphasized all the more.

In both examples, the APMN4BI models are used for an exact (unambiguous) documentation/specification of analysis processes, with two different motivations being shown (specification of a process for data quality assurance; specification of a process for discovering interesting abnormalities). Such specifications can serve as a specification for implementations (for instance, QA machine) as well as for manual analysis activities by human beings (analyst—regardless of whether they are experienced or novices). Such documentation can also be used as a means of communication to “outsiders” (for example, external consultants or external developers). In general, however, it must be assumed that the symbolism and methodology for using and reading APMN4BI are mastered. This is associated with a certain amount of learning.

This case study automatically yields an evolutionary approach. Processes were modeled, weak points and generalizations were recognized—the model was modified, expanded or generalized. In general, different versions (“intermediate versions”) and variants of analysis processes were created.

In this iterative development process, the desire for generalization was particularly evident. This wish was obvious in this case study, because due to the enormous volume of key figures and data, one tries to cover as many specific analysis steps as possible at the instance level with as few as possible at the schema level.

In both examples, tolerance intervals were determined in advance using statistical methods on the basis of historical data. Results of statistical methods can be used in APMN4BI models or APMN4BI models can define processes that provide data for static methods. However, APMN4BI cannot replace statistical methods (analytics methods).

In the case of key figure analysis, key figures were computed in advance using complex ETL processes²⁰. In APMN4BI (or in the eDFM), key figures can be defined on a “formula basis”. However, APMN4BI cannot be used to reproduce key figure calculation that are to be computed using such complex

²⁰Details and examples of such complex ETL processes for computing key figures can be collected in the LEICON projects.

ETL processes.

As with the other case studies, the need for automation and software support in the modeling and also in the execution of models must be pointed out in the present case. With a suitable modeling tool, analysis processes could be formally precisely defined and documented in a quick way. Documentation with a “universal drawing program” (for instance, Visio) is tedious. An execution tool could be used to perform data queries and analysis tasks automatically or semi-automatically.

7.2.2.2 Case Study 2: KOTI Kobra

As an evaluation objective, it was checked by way of examples whether real analysis processes from the business field of KOTI Kobra (brush production) can be modeled with APMN4BI and whether these models are also useful.

Together with specialists of KOTI Kobra, analysis processes were searched for. Existing reports, which are regularly used and analyzed for specific purposes (determining material requirements, sales control), served as a basis in order to gain new knowledge and to derive necessary actions / measures. Therefore, this analysis activity was taken from the real work environment. Hence, the analysis processes derived from these analysis activities also have a real origin.

Analysis process models are less useful for simple key figure observations (for example, monthly profit reports to be submitted to the management of the parent company). The key figures are easy to understand and can be easily interpreted at the top level (management view).

However, if you want to question profit variances more precisely, the specialists will make further inquiries based on their experience and knowledge. Certain approaches to the analysis have proven to be useful for this purpose. These procedures can be modeled as analysis processes (as in the example shown for sales control).

The example of the demand assessment for production materials shows that analysis activities are also useful to support operational processes (here from the procurement area). This requires knowledge of specialists and also

a certain routine in target-oriented approaches with respect to analysis. In turn, these approaches can be specified in analysis process models.

The benefits of analysis process models were questioned using both examples. So far, due to their routine and their knowledge, the specialists have also carried out these analysis steps and these established procedures successfully, and without difficulties. However, with APMN4BI, these analysis steps can be precisely documented. The question arose, what added value does this exact (unambiguous) documentation have.

Firstly, APMN4BI enables clear (unambiguous) documentation for “outsiders” (for instance, external consultants) and “novices” (for example, new employees who have to be trained step by step). However, for such persons, a certain effort must be invested in learning and reading APMN4BI. In addition, the experienced employees (specialists) must also learn and be able to use APMN4BI.

New employees are trained gradually—eventually, a “novice” becomes a “routinier”. The advantage of modeled analysis processes will only prove to be useful for this group of employees, if there are many different and complex analysis processes that have to be learned—usually in larger companies and organizations, provided that a “novice” is concerned with many and complex analysis processes.

It is different with “outsiders”. Consultants usually advise many companies and organizations. Although consultants have in-depth industry knowledge, company-specific features must always be viewed as a challenge. APMN4BI can serve as a suitable means of communication in the area of OLAP analyzes. A consultant (for instance, in the role of a business analyst) can learn APMN4BI once and then use it as often as required in different companies in order to precisely document a wide variety of analysis processes and reuse them for her/his work.²¹ Similar considerations can also be made in large companies and organizations. An “area” (e.g., statistics department, controlling department, parent company) can be viewed as an “outsider” to another area.

²¹Analogy: Business processes in BPMN, for example, are often also modeled by consultants in order to create documentation and, consequently, a basis for discussion.

Approaches for target-aimed and repeatable analyzes are developed step by step (“evolutionary development process”). Different versions (“intermediate versions”) and variants of approaches (analysis processes) arise (“you start and then you always get new ideas”)—this reflects the exploratory character in the data analysis. Analysis processes are modified, generalized or specialized. It makes sense to document and archive intermediate versions and alternative variants. APMN4BI would be suitable to carry out this documentation exactly (unambiguously). Later on, intermediate versions/variants can be picked up and further developed into a new and meaningful analysis process.

This evolutionary approach is ultimately also used when developing reports. A first report is created, it is used, new wishes and requirements arise and the report is further developed. Such reporting is usually based on dynamic reports (filter options, drill options, links). APMN4BI could also be used as a specification for a reporting system.

Although not implemented, useful and necessary options for automation based on APMN4BI (including eDFM) can be recognized. By a suitable modeling tool, analysis processes could be formally precisely defined and documented in a quick way. Documentation with a “universal drawing program” (for example, Visio) is tedious. An execution tool could be used to perform data queries and analysis tasks automatically or semi-automatically.

7.2.2.3 Case Study 3: PVA

As an evaluation goal, the two use cases from the pension insurance institution—in German: Pensionsversicherungsanstalt (PVA)—were used to check whether real analysis processes from the rehabilitation and pension area of the PVA can be modeled with APMN4BI and whether these models are also useful. The analysis examples considered in this case study were taken from a real, current application context of PVA with current objectives: (1) Planning of future rehabilitation facilities (“Rehab 2022”) with a strategic focus on outpatient rehabilitation; (2) Long-term rehabilitation evidence (with the effect on disability pensions and rehabilitation money).

A special feature of this case study was that these analysis examples were themselves just in the development stage. This means, no analysis processes were considered that are carried out using reports or cubes that have existed for years and, therefore, that have proven themselves with respect to functionality and data, but data cubes, reports and dashboards, and other visualization/interaction variants (interactive map) were just developed. With this special feature, two beneficial aspects of APMN4BI could be demonstrated:

(1) As a specification of analysis processes, APMN4BI can also be viewed as a specification for reports, dashboards or other forms of presentation that is independent of the visualization: The specification of an analysis process defines requirements for the dashboard, et cetera! This supports consulting activities, in particular, business analysis and requirements engineering activities.

(2) APMN4BI supports the evolutionary (iterative) approach for data analysis. Concrete analysis processes are documented in APMN4BI (also in the sense of “logging”, “archiving”, or in the sense of “don’t forget”) and discussed with the department. Suggestions for improvement obtained from this are incorporated into APMN4BI models—analysis processes are adapted or expanded in APMN4BI, or new/additional analysis processes are defined. Often, from very “concrete analysis processes” (“instance-related”, with no or only a few variables)—usually again from technical discussions—general analysis processes (“schema-related”) are derived.

The evolutionary procedure is not limited to the analysis processes themselves. Data structures (eDFM) and data (via ETL) must also be further developed in parallel.

The example of the interactive rehabilitation map shows that APMN4BI allows analysis processes to be represented independently of visualization. The results of analysis situations are used for the visualization. The structuring into the layers “Analysis Process Layer” and “Presentation & Action Layer” and the connection between them is recognizable and makes sense. As a conceptual modeling language, APMN4BI is limited to the “Analysis Process Layer”.

In order to increase the expressiveness of APMN4BI, the following exten-

sions should be aimed for. On the one hand, expanding the eDFM to include entity types, dimension hierarchies and dimension roles would simplify the conceptual modeling of the underlying data and make it easier to understand. On the other hand, with an extension of the `drillAcrossToCube`-operator, in which slice conditions are introduced, the results from the start analysis situation could be transferred as a filter to the target analysis situation, and further classes of tasks could be implemented in APMN4BI.

As already shown in the two other case studies (KOTI Kobra and LEICON), suitable software support is essential for creating eDFM and APMN4BI models. With a suitable modeling tool, such models must be able to be created, modified and extended quickly. Using a “universal modeling and drawing program” such as, in this case, Microsoft Visio, is labor-intensive and inefficient to create eDFM and APMN4BI models. Especially with an evolutionary (iterative) approach, the use of a modeling tool specifically designed for eDFM and APMN4BI is essential.

7.2.3 Consolidated Evaluation

This section refers to the final case studies presented in [87, 88, 89] but also to other use cases for the case studies’ environments. The goal of this section is to provide a consolidated evaluation over all case studies and use cases such that all main constructs of APMN4BI are examined at a coarse level. Thus, in the following subsections, non-comparative analysis situations and eDFM’s, comparative analysis situations, navigation steps and navigation operators, derived cubes, extensions to schema level, and the organization of BI analysis graphs are evaluated in the context of real case studies and use cases.

7.2.3.1 Non-comparative Analysis Situations and eDFM’s

A non-comparative analysis situation represents a fundamental conceptual construct for business analysts to query data based on multi-dimensional data models. The constituents of an eDFM comprise common elements of a common DFM at a conceptual level: a cube comprises dimensions and base

measures; dimensions consist of hierarchies of dimension levels that optionally can contain descriptive attributes. Additionally, the DFM is enriched by further conceptual constructs: base measure predicates, aggregate measures, aggregate measure predicates, scores, score predicates, dimensional predicates, dimensional operators, and join conditions. Non-comparative and comparative analysis situations are constructs that use these constituents of an eDFM to define non-comparative and comparative queries at a conceptual level based on multi-dimensional data models. In this subsection, non-comparative analysis situations are focused.

In the case of public health insurance organizations, we found many multi-dimensional cubes to be queried. These cubes are provided as star schemas. Cubes or star schemas for drug prescriptions, ambulant treatments, and hospitalizations are only three examples. Also for the manufacturing use cases, cubes are provided basically as star schemas (ROLAP based) but also, additionally, as multi-dimensional Cognos cubes (MOLAP). During the case study of the pension insurance organization, multi-dimensional data models just were developed. In this case study, cubes based on a star schema were elaborated but there also exist many simple flat tables that have to be used for multi-dimensional analysis. These examples showed that an eDFM provides a conceptual view of multi-dimensional data independent from real implementations as star schemas, MOLAP-cubes, or simple flat tables.

The enrichment of a DFM by base measure predicates, aggregate measures, aggregate measure predicates, scores, score predicates, dimensional predicates, dimensional operators, and join conditions was a novelty in that, previously, such elements only were considered as plain technical expressions, whereas now, in APMN4BI, they obtain a name and become elements of an eDFM such that they can be considered as conceptual constructs that can be used by business analysts. The abstraction as named elements (named dimensional predicates, named aggregate measures, etc.) and the organization of such elements in hierarchies (for example, a hierarchy of dimensional predicates or a hierarchy of aggregate measures) caused positive acceptance, although, in our case without tool support, these concepts only applied for documentation purposes. The semantics of queries increase by such concep-

tual constructs which can also be regarded as a kind of business metadata (for instance, in the sense of a business glossary).

Currently, with respect to the environments of the case studies, measures and predicates are often pre-computed by ETL processes and stored in table attributes. With appropriate tool support based on APMN4BI, the definition of measures and predicates can be done by business analysts themselves, which would provide more flexibility.

Based on use cases, we recognized that defining dimensional predicates instead of dimension levels provide advantages in many cases. Dimensional predicates are preferable to dimension levels, especially in the case of “artificial” dimension levels. As an example, we consider various age groups that are often realized by table columns. Age groups of patients often depend on the analysis question. For DM2 patients, other age groups are of interest than for analysis of healthcare for children and youth. A general dimension hierarchy with age groups divided in decades and age groups divided in five steps is not necessarily appropriate. Moreover, such age groups often are not balanced, for example, for DM2, finer gradation is needed for older patients than for younger ones. Also in the case of brush manufacturing, a hierarchical categorization of, for example, trimming diameters for bristles is difficult to be mapped on “artificial” dimension levels. Dimension predicates and predicate hierarchies do not claim a strict leveled hierarchy and thus provide more flexibility.

For querying multi-dimensional data, non-comparative analysis situations provide a conceptual construct for business analysts. A non-comparative analysis situation refers to a cube that is based on an eDFM. Data selection can be restricted by base measure predicates (base measure conditions), by dimension nodes at specific dimension levels (dice node and dice level), and by dimensional predicates (slice conditions). Measure values are aggregated with respect to granularity levels and the result can be additionally filtered by aggregate measure predicates. Dice levels, dice nodes, slice conditions, and granularity levels refer to specific dimensions, and can be considered as a dimension qualification. The constituents of a non-comparative analysis situation provide the component of an SQL query that can be used to retrieve

the desired data.

Non-comparative analysis situations can be found in all three case studies presented in [87, 88, 89]. An analysis process often starts by a non-comparative analysis situation. Later in the process, analysis situations appear to provide comparison (comparative analysis situations). For example, in the LEICON case study all presented analysis processes are started by a non-comparative analysis situation: calculation of total drug prescription costs of a specific year or calculation of total prevalence of DM2 patients. Also the analysis processes of the KOTI Kobra case study and PVA case study have non-comparative analysis situations as an entry point.

At this place, note that BI analysis graph schemas have no “start analysis situation schemas”. Nevertheless, the analysis process itself mostly starts at a predestined analysis situation schema (in many cases a non-comparative analysis situations schema) that is used as a kind of “start analysis situation schema”. At instance level, such a “start analysis situation schema” is used to generate the root of a BI analysis graph that represents a specific analysis process.

The case studies reveal that the eDFM itself can be considered as a valuable conceptual construct. Especially, the enrichments provide additional semantics for a DFM. As a result of the gained insights of the case studies, all enrichments were adapted as named constructs. Especially, also join conditions are part of an eDFM that obtain a unique name. Although not used in this thesis, there are no impediments to introduce inline definitions without names.

Moreover, the case studies showed that it would make sense to introduce reusable elements into an eDFM like entity types. Dimension levels (for example, districts or provinces) are used in different dimension schemas but always have the same descriptive attributes like population. Once defined as entity types, they could be used several times in various dimension schemas.

Another suggested improvement identified in the case studies was to use name spaces for dimensions. Dimension levels and descriptive attributes could be encompassed by name spaces such that equal names can be used in more than only one dimension. Moreover, one could also introduce dimension

roles to improve reusability of dimensions. Especially in the PVA case study, the reusability of dimension hierarchies and, furthermore, the reuse of sub-hierarchies in another cube was an additional desire. For simplicity with respect to formalize APMN4BI, we refrain from such concepts like entity types, name spaces, and dimension roles.

The LEICON and KOTI Kobra case studies also showed that the use of multi-ary predicates is an additional feature which is essential towards universal use of APMN4BI. Although, for simplicity, we only introduced dimensional predicates, base measure predicates, aggregate measure predicates, score predicates, and join conditions as 0-ary predicates, the concepts of APMN4BI can also be extended to n-ary predicates. Operators as defined in Table 5.1 in the context of navigation guards return the constituents of analysis situations and could be used more generally, especially to serve as parameter values for multi-ary predicates. Furthermore, in the LEICON case study, parameterized predicates and functions were used that can be considered as an access to lookup tables to obtain values which had the advantage to define predicates more general and to use them in a more universal manner such that analysis processes could be modeled more convenient and simpler.

In the LEICON case study, variables were used to increase generalization of analysis processes. For example, it became apparent that also variables for measures are useful such that at schema level many analysis situation schemas could be saved. Hence, the use of variables for all components of a non-comparative and also for a comparative analysis situation was incorporated later in this thesis after finishing the final case studies described in [87, 88, 89]. The only non-variable item of a non-comparative analysis situation represents the cube schema; but also cube instances can become variable. For these changes towards the possibility of an extensive use of variables, a main part of the formalization of APMN4BI had to be adapted.

Finally, the presentation of an eDFM was revisited and adapted on the basis of the insights gained in the final case studies summarized in [87, 88, 89]. Derived base measures, join conditions, scores, and score predicates become formal as well as visual members of an eDFM.²² Therefore, all data struc-

²²Originally, join conditions, scores, and score predicates were introduced for compar-

tures of an enriched multi-dimensional data model necessary for APMN4BI are completely contained in an eDFM. Section 2.4 comprises the new definitions of an eDFM that also contain improvements gained from the insights of the case studies. Especially, a clarification of the notions base measure, simple base measure, derived base measure, aggregate measure, simple aggregate measure, and derived aggregate measure were introduced formally and visually such that one can recognize that the definitions of these notions are based on each other.

7.2.3.2 Comparative Analysis Situations

A comparative analysis situation sets the focus on a multi-dimensional query represented as a non-comparative analysis situation (the context of interest) and compares it with the result of another multi-dimensional query, again represented as a non-comparative analysis situation (the context of comparison). Such a comparison represents an activity of a business analyst. She or he focuses some measures within a certain context (context of interest) and compares them with measures of another meaningful context (context of comparison)—meaningful in the sense of that both contexts are related appropriately (by join conditions). To measure similarities between the measures of both contexts, she or he calculates scores. Depending on the score values, further decisions are made with respect to further analysis, i.e., a business analyst filters compared rows with respect to filter conditions.

Such comparing activities of business analysts could be observed in many use cases of the environments selected for the case studies and introduced in Section 1.3. For instance, patients of a province are compared with patients of another province, DM2-DMP doctors are compared with Non-DM2-DMP doctors, the revenue of a month is compared with the revenue of same month of the previous year, production costs of brushes with a certain trimming diameter for bristles are compared with brushes of another trimming diameter. In these cases, it is useful to document such sometimes tacit comparisons of

ative analysis situations but independently from an eDFM despite they were based on them. Derived base measures were not depicted in an eDFM digram but only served as a constituent to define aggregate measures.

a business analyst explicitly.

In the LEICON case study presented in [87], drug prescription costs, hospitalization costs, and many other measures of a year are compared with the previous year. The KOTI Kobra case study comprises an analysis process where the revenue of a month is compared with the revenue of the same month of the previous year [88]. The second analysis process of the PVA case study in [89], contains comparative analysis situations that compare insurants that obtain occupation disability insurance with and without a preceding rehabilitation.

The KOTI Kobra and the PVA case studies in [88, 89] also contain analysis processes without comparative analysis situations at all. This does not necessarily mean that a business analyst performs no comparisons but comparison is not realized by comparative analysis situations. For example, one analysis process of the KOTI Kobra case study without comparative analysis situations has non-comparative analysis situations with appropriate granularity levels that list quantities per material, per product, per quarter, and per customer, respectively. Having a list of products with aggregated quantities of a certain material, of course, a business analyst would compare the various rows of that list. But she or he does not use comparative analysis situations for these comparisons.

Another insight gained from the case studies was to introduce named join conditions to an eDFM. Every join condition is defined as a DFM enrichment and obtains a name. Thus, a comparative analysis situation comprises a join condition by its name. Also “standard joins” (equi joins with respect to granularity levels) can be used in this way. Additionally, we obtain a uniform treatment of conditions and predicates. All conditions and predicates (i.e., join conditions, base measure predicates, aggregate measure predicates, score predicates, and dimensional predicates) are defined in an eDFM and used in analysis situations by their names. Named join conditions were incorporated later in this thesis after finishing the final case studies presented in [87, 88, 89].

7.2.3.3 Navigation Steps and Navigation Operators

A navigation step shows how an analysis situation is modified to retrieve another analysis situation. Navigation from one to another analysis situation is based on the knowledge and experience of business analysts. The difference of both analysis situations is made visible by the navigation step.

Although business analysts perform OLAP operations like drill down or roll up, these invocations are not documented explicitly in a common OLAP tool. By APMN4BI, such operations can be described and documented in a precise way. Moreover, the repertoire of operators exceeds the repertoire of most OLAP tools. Additional semantics is expressed by navigation operators of APMN4BI.

All navigation operators presented in Chapter 4 (non-comparative as well as comparative navigation operators) were elaborated over many years on the basis of use cases of real environments introduced in Section 1.3. Understandability and usability was a main focus for specifying and designing navigation operators. Also the visual design of the operators' pictograms was adapted as long as a clear symbolism was found such that the behavior of an operator could be interpreted in a comprehensible and consistent way. In spite of it all, a certain learning curve is required to understand and apply these navigation operators. Especially, users have to be familiar with multi-dimensional data models as presented in a conceptual manner by an eDFM.

The final case studies in [87, 88, 89] comprise at least one example per non-comparative and comparative operator category: non-comparative operators changing granularity level, dice node, slice conditions, base measure conditions, aggregate measures, filter conditions, and cube access, and comparative operators introducing comparison, changing comparison, and dropping comparison. These case studies together with all use cases found in real environments demonstrated that the history of invocation of OLAP operations can be documented sufficiently by navigation steps and navigation operators. The reproducibility and comprehensibility of documented analysis processes was observed as a main advantage.

In the subsequent paragraphs, suggested improvements concerning nav-

igation steps and navigation operators, and gained from the insights of the final case studies presented in [87, 88, 89] are explained. Some of these improvements were incorporated in this thesis after the completion of the case studies, others were not incorporated for simplicity.

One suggested improvement concerns sequencing of navigation operators. The use of a single navigation operator often does not result in a meaningful target analysis situation from the business view. In such cases, a meaningful target analysis situation can only be obtained by executing several operators one after the other. This requirement was observed in the LEICON and KOTI Kobra case study, as well as in the PVA case study. Nevertheless, for the sake of simplicity, we refrain from formalizing sequences of navigation operators that have no analysis situations between them. In the case of operator sequences, one can think of “pseudo” analysis situations that are positioned between two navigation operators of an operator sequence and that are not visible. Hence, the missing of an appropriate formal construct for operator sequences does not cause serious restrictions for APMN4BI. The PVA case study additionally demonstrates operator sequences containing a fork. In this example containing three target analysis situations, the branching point can be considered as a special “pseudo” analysis situation which represents a source analysis situation from where three navigation steps lead to three targets. In other words, one can consider three different operator sequences.

A fundamental change that was elaborated after finishing the final case studies and that was at the basis of the gained insights of these case studies concerns navigation guards. In the first version of APMN4BI, navigation guards were defined as an add-on for navigation step schemas. Although in this first version, on one side, navigation guards were applied at instance level to examine the result set of the query of a source analysis situation and, on the other side, navigation guards were also applied to examine variable assignments in source analysis situation schemas when instantiating them, the case studies revealed that navigation guards are formally better located as an inherent constituent of navigation steps at instance level. Thus, navigation guards become a constituent of the definition of a navigation step. A

navigation guard represents a boolean expression that, if evaluated to true, allows to perform a navigation step, i.e., the subsequent navigation operator is invoked, otherwise the navigation step does not lead to the target analysis situation. With respect to the second case, the query of the target analysis situation is not executed and all subsequent navigation steps that have this target as source are also not performed. In the case, a navigation guard is defined by the constant boolean expression `true`, we have the situation that the navigation step is always executed. This could also be interpreted as that there exists no “explicit” navigation guard at all, although, formally every navigation step has a navigation guard per definition (in this case, just the boolean constant `true`). Summarizing, also in the new formalism, we can distinguish two main purposes of navigation guards: First, navigation guards can be used to examine result sets after query execution. This can only be done at instance level. Secondly, navigation guards can be used to examine variable assignments in source analysis situation schemas. This refers primarily to schema level at the point of instantiation, although, the evaluation of the boolean expression and the navigation guard itself belong also to the instance level.

Other insights gained from the case studies in [87, 88, 89] concern the navigation operators themselves. On the one side, the visual design was improved²³, on the other side, new variants of operators were introduced for changing iteratively dice nodes of an arbitrary sub-level of a dimension qualification of an analysis situation. Thus, new variants of operators `moveDownToFirstNode`, `moveDownToNextNode`, `moveDownToLastNode`, and `moveDownToPrevNode` were introduced that do not only affect the direct sub-level but that can also affect arbitrary (also indirect) sub-levels. This improvement was inspired by the KOTI Kobra case study.

Another suggested improvement from the PVA case study was not incorporated formally in this thesis for simplicity. It concerns the `drillAcrossToCube` operator. Often it would be valuable, if the result of an analysis

²³This concerns improvement of the visual representation of pictograms, the uniform use of pictograms with respect to the eDFM and the actual parameters, and the use of operator names with a unique meaning, for example, `narrowSliceCondition+` versus `narrowSliceCondition->`.

situation can be used in the target analysis situation to narrow the set of dimension nodes (in the sense of a dimensional predicate). For example, if an analysis situation determines the set of patients with rehabilitation, then the `drillAcrossToCube` operator would change the cube (from the rehabilitation cube to the pension cube) and the target analysis situation would select from this set of patients the ones that obtain an occupational disability pension.

Other aspects that were evaluated in the case studies concern the use of variables as actual parameters for navigation operators. If using variables with names, the operator invocation is better interpretable. On the other side, it was also useful to have specific operators that return single constituents of a source analysis situation. These operators are excessively applied in the context of navigation guards. Furthermore, the navigation operators' actual parameters were depicted in a more intuitive way: simple notation referenced by a name where the construct is defined including a unique name in the eDFM; the appropriate symbol in the eDFM is also used as a visual prefix for the actual parameter. Also standard joins (equi joins) represent a defined element of the eDFM such that the name is used as an actual parameter. Moreover, in the PVA case study, an operator call was demonstrated that refers different dimension roles as an actual parameter.

7.2.3.4 Comparative Navigation Operators

Whereas navigation steps from a non-comparative to another non-comparative analysis situation were rather familiar for users, the navigation to comparative analysis situation took up more elaboration and evaluation cycles. As a result of this process we obtained the comparative navigation operators `relate` and `target` as defined in 4.3.1.

The idea for comparative operator `relate` is due to the inherent intention of comparative analysis situations. A business analyst takes a look on a non-comparative analysis situation (context of interest) and intends to obtain new insights by comparing it with another non-comparative analysis situation (context of comparison). The relationship between the context of interest

and the context of comparison is specified by a non-comparative navigation operator. It expresses how one comes from the context of interest to the context of comparison. If we are interested in drug prescription costs of Upper Austria, one would like to compare them with drug prescription costs of Austria, or if one is interested in the revenue of a month, she or he would like to compare it with the revenue of the same month of the previous year.

Sometimes a business analyst intuitively starts with the context of comparison. She or he watches the drug costs of Austria and is interested in the next step to look at the drug costs of Upper Austria. Similarly, users looked at revenues of a month of the previous year and changed their view to the month of the actual year. The operator `target` provides this activity.

Similarly, operators `rerelate`, `retarget`, and `correlate` were elaborated and evaluated with respect to comprehensibility and usability in this way. To focus on one context (either the context of interest or the context of comparison), we introduced operators `unrelate` and `untarget`. Operators `rejoin` and, for changing score filters, `narrowScoreFilter+`, `narrowScoreFilter->`, `broadenScoreFilter-`, `broadenScoreFilter->`, `refocusScoreFilter`, and `refocusScoreFilter->` were introduced for completeness to provide flexible options for changing constituents of comparative analysis situations.

Most often, comparative navigation operators are used that have to be combined with non-comparative navigation operators. The non-comparative navigation operators define how the context of interest and the context of comparison are derived from a non-comparative source analysis situation, or how the context of interest and/or the context of comparison of a comparative source analysis situation are changed. Comparative navigation operators that eliminate the context of interest or the context of comparison of a comparative source analysis situation, or that only change constituents of a comparative source analysis situation which do not belong to the context of interest and the context of comparison are not combined with non-comparative navigation operators.

In the case studies presented in [87, 88, 89], four analysis processes comprise navigation steps with comparative navigation operators and two analysis processes only contain navigation steps with non-comparative navigation

operators. The design and behavior of non-comparative navigation operators were elaborated over many years. The final case studies only caused minor adaptations concerning operator names, depiction of parameters, and slight visual adaptations. One later improvement, for instance, relates to the use of join conditions by name that are defined in the underlying eDFM.

7.2.3.5 Derived Cubes

In a later elaboration and evaluation cycle, we recognized the need of derived cubes and the need for navigation operator `useAsCube`. In some use cases, there exists the situation that a query result comprises already aggregated values that have to be considered as base measures for further aggregation. In this case, we interpret this query result again as a multi-dimensional cube induced from an analysis situation.

For instance, for DM2 analysis, business analysts were not interested in average costs over all individual drug prescriptions but they were interested in the average over the yearly cost sums of drug prescriptions per patient. In this case, the yearly costs of the drug prescription cubes have to be summed up per patient and, afterwards, the cost sums per patients can be used to calculate the average costs over all patients which again can be grouped by other dimension levels like provinces. There were also use cases in the brush manufacturing environment. For instance, firstly, the yearly revenue sum was computed per customer and, afterwards, the yearly revenue sums per customer were used to calculate the average revenues over all customers. Again, this can be grouped by, for example, regions.

Such requirements are often implemented by additional ETL steps that stores already aggregated data to proceed with further aggregations. As an alternative, appropriate view definitions instead of storing data would lead to the same result. Conceptually, this stored data (or view definitions) are considered as derived cubes as introduced in Section 4.4.1. The operator `useAsCube` derives cubes from a source analysis situation and accesses this derived cubes in the target analysis situation and in subsequent navigation steps. Conceptually, such a derived cube is again represented at schema level

by an eDFM.

The final case studies in [87, 88, 89] do not comprise examples that demonstrate the usage of the `useAsCube` operator. As previously mentioned, one reason was that such derived cubes already exists as “common cubes” that were generated by separate ETL processes. In this case, instead of using the `useAsCube` operator that introduces a derived cube, one can use the `drillAcrossToCube` operator to change to another cube that already contains precomputed aggregations. Nevertheless, at a conceptual level, it would be more transparent to model such analysis processes by the `useAsCube` operator.

A similar use case where the query result of a source analysis situation is used to execute the query of the target analysis situation was presented in the PVA case study. In this case study, a source analysis situation determines insurants that had rehabilitation and from this set of insurants the ones are selected that, afterwards, obtained an occupational disability pension. The difference is that such query result sets are not used as cubes with coarser base levels (derived cubes) but they are used to restrict the selection of dimension nodes in the target analysis situations. Thus, one can think of a kind of dimensional predicate that uses dimension nodes from the result set of the source analysis situation to restrict the dimension nodes of the target analysis situation. In this thesis, we refrained from formalizing such dimensional predicates to reduce complexity of an eDFM.

7.2.3.6 Extensions to Schema Level

During the elaboration of APMN4BI for years, a clear and uniform distinction between schema and instance level was obtained. In earlier stages of APMN4BI, schema and instance elements existed in a hybrid manner. A main feature at schema level represents the use of variables for constituents of analysis situations and for actual parameters of navigation operators. One purpose of schema elements concerns generalizations. Whereas at instance level, a specific analysis process is documented comprising specific and directly executable queries, at schema level a class of analysis processes is

described.

In the final version of APMN4BI, all constructs at instance level correspond to a construct at schema level that generalizes the construct at instance level or, looking at it the other way round, a construct at schema level can be used to instantiate a specific construct at instance level. An eDFM defines conceptually a cube schema that contains dimension schemas. A cube represents an instance of a cube schema. It comprises dimensions which are instances of the corresponding dimension schemas. Whereas cube schemas and dimension schemas can be associated with table schemas, cubes and dimensions are related to database tables containing concrete data.

Analysis situations represent specific queries that can be translated to SQL statements which refer database tables and which can be executed directly as database queries. An analysis situation schema is used to generate, in general, more than one analysis situation, i.e., an analysis situation schema induces, in general, more than one database query. Variables are used to abstract from analysis situations and to generalize analysis situations to analysis situation schemas. Analysis situation schemas without variables directly represent analysis situations where no variable assignments are necessary. This corresponds to a special case of analysis situation schemas. In this case only one analysis situation can be generated from the analysis situation schema.

Similar to analysis situations, navigation steps are generalized to navigation step schemas. Again, the use of variables makes it possible to define navigation step schemas from which more than one navigation steps can be instantiated. Variables in navigation step schemas concern variables in the source and/or target analysis situations, but also variables used for actual parameters in navigation operators. Similar to analysis situations, the definition of navigation step schemas without variables represents a special case from which only one navigation step can be instantiated. Navigation guards can be considered as constituents at schema level as well as constituents at instance level. In general, the evaluation of navigation guards is only performed at instance level. To determine the moment of type checking (type checking at schema or at instance level), the notion of type-compliant (i.e., schema-compliant) navigation steps was introduced (see Section 5.2.3) and

the prerequisites for static type checking (type checking at schema level) and dynamic type checking (type checking at instance level) were discussed.

Finally, the extension to schema level was transferred to analysis graphs. BI analysis graph schemas represent directed multi-graphs from which, in general, more than one BI analysis graphs can be instantiated. This also concerns subgraphs, especially, composite analysis situations. BI analysis graphs represent directed trees. The visualization of analysis situation schemas, the visualization of subgraphs at schema level as demonstrated in Section 6.4, and the visualization of composite analysis situation schemas are clearly distinguished from the corresponding constructs at instance level. At schema level, double-edged shapes are used, whereas at instance level, we use single-edged borders.

In all case studies of [87, 88, 89], eDFM's and BI analysis graph schemas were defined at schema level. This corresponds to modeling activities and represents actually the creative part of business analysts. But this does not mean that analysis situations and navigation steps are not taken into account. On the contrary, modeling of analysis processes in APMN4BI represents an iterative and evolutionary process where instances of analysis processes are studied, afterwards, model fragments at schema level are created, afterwards, again, instances of such gained model fragments are examined, possibly further navigation steps at instance level are analyzed, again, further model fragments at schema level are generated, and so on. In the end, the final APMN4BI models at schema level only represent the final product of this modeling process. Exemplary instances of the BI analysis graph schemas were not depicted in [87, 88, 89]. Nevertheless, exemplary instantiations were performed together with users to demonstrate the meaningfulness and the reusability of developed APMN4BI models. Finally, such models can be used to document concrete analysis processes at instance level.

One difficulty in the performed case studies was that there was no modeling and execution tool for APMN4BI available. A simple drawing program like Microsoft Visio only allows a very simple and laborious modeling process. For APMN4BI, it is indispensable to develop and provide an appropriate modeling tool. Nevertheless, simple drawing programs can at least be used

to create a static documentation of analysis processes.

Whereas the LEICON and KOTI Kobra case studies were designed to model existing analysis processes that users perform periodically without APMN4BI, the PVA case study showed an example where the analysis process itself was elaborated iteratively by using APMN4BI. This demonstrated that APMN4BI can be used to develop and refine new analysis processes provided one has an appropriate modeling tool.

Another insight that was gained from the case studies refers to navigation guards. As already mentioned in Section 7.2.3.3, in earlier versions of APMN4BI, navigation guards represented an exclusive part at schema level that were considered as add-ons. Now, in the final version of APMN4BI, navigation guards are inherent elements of navigation step schemas as well as of navigation steps.

7.2.3.7 Organization of BI Analysis Graphs

The representation and invocation of BI analysis processes by BI analysis graph schemas and instances was the overall result of the elaboration and evaluation process. Examples of analysis processes were collected in all three environments for case studies as introduced in Section 1.3 and modeled by APMN4BI. Use cases from Austrian public health insurance organizations were taken from analyzing diseases as sketched in Section 1.3.1, especially, analysis of DM2 patients, evaluation of DM2 DMP's, but also other subjects like, e.g., health care of children and youth, mental illness, patients with chronic heart failure, early detection of breast cancer. As briefly described in Section 1.3.2, in the case studies about brush manufacturing, monthly analysis of revenue and profit, analysis of ordered and used material, and analysis about the production process provided use cases for APMN4BI and its evaluation. Finally, in the area of Austria's public pension insurance organization (see Section 1.3.3), use cases for planning new rehabilitation facilities and use cases to provide evidence of effectiveness of rehabilitation with respect to retirement were found and examined in more detail.

In all three application areas, the need of organizing BI analysis graph

schemas and BI analysis graphs arose. BI analysis graph schemas can be organized with respect to goal hierarchies (see Section 2.3). One BI analysis graph schema solves one analysis task, which can again be decomposed. The usage of subgraphs provides simple but sufficient means of decomposing BI analysis graph schemas and BI analysis graphs (see Section 6.4). A specific need was detected in the sense that some analysis situations have to be considered and also visualized simultaneously. This inspired the definition of composite analysis situations as presented in Section 6.5.

All analysis processes depicted in the case studies of [87, 88, 89] can be considered as subgraphs because, in general, they are embedded in more comprehensive analysis processes.²⁴ On the one side, parallel to an analysis process another analysis process is defined where both are subprocesses of a more comprehensive one, on the other side, an existing analysis process can be refined such that various subprocesses arise.

For instance, in the LEICON case study, after defining the analysis process for performing quality assurance of drug prescription costs, further analysis processes were defined for executing quality assurance of ambulant treatment costs, hospitalization costs, and for transportation of patients. All analysis processes can be considered as subprocesses that are embedded in an overall analysis process for quality assurance. This corresponds to a bottom-up approach that generates BI analysis graph schemas that are embedded as subgraphs in an overall BI analysis graph schema.

In the second analysis process of the KOTI Kobra case study, first the analysis process was sketched in a coarse level and, afterwards, subprocesses were refined. For example, the analysis task to compare the monthly revenue with the calendar month of the previous three years was indicated as a subprocess of the whole analysis process and, afterwards, this subprocess was specified in detail in a second step. Such an approach corresponds to a top-down proceeding that refines a BI analysis graph schema that comprises further BI analysis graph schemas as subgraphs.

Subprocesses also arose as a consequence of an iterative and evolutionary

²⁴Note, not every analysis process of the case study is depicted in the final report presented in [87, 88, 89].

modeling approach. In the second modeling iteration of the first analysis process of the LEICON case study, the APMN4BI model was adapted in such a way that the quality assurance (deviation analysis) of a certain drug group (accordingly to ATC classification) and with respect to a certain insurance company was generalized by introducing additional variables. These generalized analysis steps were encapsulated in an additional subprocess which was embedded in the whole analysis process.

Most subgraphs depicted in case studies of [87, 88, 89] were defined as composite analysis situation schemas. Although there was no runtime environment available to execute an instantiated composite analysis situation in one go²⁵, the encapsulation into composite analysis situation schemas made sense anyway because a complex BI analysis graph schema could be structured into subgraphs and thus, a better overview was obtained. Furthermore, it is recognizable that the queries of all analysis situations that belong to a composite analysis situation should be executed coherently. In this sense, composite analysis situations and composite analysis situation schemas also provide additional semantics.

7.3 Evaluation of Claimed Design Criteria

This Section presents arguments about the fulfillment of the design criteria presented in 1.4.2. These design criteria were chosen to satisfy the aims presented in 1.4.1. In Table 1.1 of Section 1.4.2, the methods are listed that are used to evaluate the corresponding design criteria.

7.3.1 Criteria 1: Domain Specific Conceptual Modeling Language

As a domain specific language, APMN4BI has to offer a conceptual notation for modeling BI analysis processes on the basis of multi-dimensional cubes and OLAP operations. In 1.4.2, we mentioned that such a language has

²⁵This represents one important feature of composite analysis situations.

to provide a close one-to-one correspondence between user conceptualization and model representation such that technical details are abstracted.

This design criteria was elaborated and evaluated in discussions and interviews with business analysts and subject matter experts on the bases of use cases. These use cases were extracted from the case studies presented in this thesis. They come from three different application areas (health insurance organizations, brush manufacturing, and public pension insurance). Hence, this emphasizes a certain universality of the application of APMN4BI. Most concepts and constructs of APMN4BI are also included in the final case studies presented in [87, 88, 89]. Final improvements based on these case studies were made in a final elaboration cycle of APMN4BI.

The general application domain of APMN4BI concerns BI analysis processes on multi-dimensional cubes and OLAP operations. Non-comparative analysis situations correspond to a multi-dimensional query. BI users are familiar with such a type of queries. Non-comparative analysis situations comprise all constituents at a conceptual level to select data from a multi-dimensional cube: the cube itself, the measures applied (including aggregation), filters with respect to the base measures stored in the cube, selection to dimension nodes that have to be aggregated, the granularity of aggregation per dimension, and the final option to filter with respect to resulting measure values. A dimension qualification comprises all dimension specific constituents (dice node with its dice level, slice condition, and granularity level). These multi-dimensional queries are visualized such that each constituent is associated with an appropriate symbol. In all three application areas, the constituents and the associated symbols were examined and adapted with respect to their usability and understandability.

OLAP operations are represented by navigation operations from non-comparative to another non-comparative analysis situation. Navigation operators are provided to manipulate all constituents of non-comparative analysis situations: changing granularity levels, changing dice nodes, changing slice conditions, changing conditions for filtering the result set, changing base measure conditions, changing measures to calculate, or even changing the underlying cube itself. OLAP operations like drill down, roll up, slice,

dice, or drill across can be performed. Moreover, the set of APMN4BI navigation operators provides more semantics and expressiveness. For instance, a user can express whether to narrow or broaden a slice condition, or iterative navigation intentions can be expressed, for example, by operators `moveDownToFirstNode` and `moveDownToNextNode`, or the navigation from one to another analysis situation can be controlled by navigation guards. Finally, all such operators and constructs are visualized by an expressive symbolic.

Beside performing OLAP operations, a main activity of business analysts represents comparison. Of course, comparison is based on several and often by many data items, and it is supported by appropriate visualization. Independently of the amount of data to be compared and independently from visualization methods, from a process view, we could observe that the “human eye” always performs the following steps: focus a data item you are interested in, relate it to another one you like to compare it with, and score both to express their similarity or dissimilarity. This observation led to the definition of comparative analysis situations. Comparative analysis situations comprise a context of interest that is related by a context of comparison both represented by non-comparative analysis situations. The linkage is given by a join condition and the similarity is measured by one or more scores. Furthermore, it was useful to introduce additional conditions for filtering the result set with respect to the score values.

With regards to model analysis processes, it was necessary to introduce navigation operations that relate a non-comparative analysis situation as context of interest to another non-comparative analysis situation as context of comparison leading to navigation operator `relate`. On the other hand, one can also start from a general analysis situation as context of comparison and navigate to an interesting analysis situation as context of interest by operator `target`. The difference between the non-comparative source analysis situation and the appropriate deviating context of the comparative target analysis situation is expressed by a non-comparative navigation operation that itself can be considered as a parameter to operators `relate` and `target`. Navigation operators `rerelate`, `retarget`, `correlate`, and `rejoin` change a

comparative analysis situation itself: changing the context of comparison by `rerelate`, changing the context of interest by `retarget`, changing both by `correlate`, or changing no context but only join conditions, scores, and/or score filters by operator `rejoin`. For only changing score filters, we introduced additional operators `narrowScoreFilter+`, `narrowScoreFilter->`, `broadenScoreFilter-`, `broadenScoreFilter->`, `refocusScoreFilter`, and `broadenScoreFilter->`, and, for extracting the context of interest or the context of comparison, operators `unrelate` and `untarget` were defined.

Non-comparative and comparative analysis situations represent elementary constructs of APMN4BI. Thus, these constructs are also contained in the final case studies of [87, 88, 89]. Most analysis processes of the final case studies contain comparative analysis situations. But there are also two analysis processes (one in the KOTI Kobra and the other one in the PVA case study) without comparative analysis situations at all.

To reduce complexity of formal definitions, sequences of navigation operators were not incorporated explicitly in APMN4BI, although they appeared in all three case studies. Nevertheless, operator sequences can be simulated by invisible analysis situations as mentioned in Section 7.2.3.3.

Although not included in the final case studies, in many other cases, we could observe that the result of analysis situations are used again for other analysis situations. This need led to the definition of derived cubes and to the definition of navigation operator `useAsCube`. An analysis situation itself induces a cube derived from the cube contained in it. Currently, such requirements are shifted to technical solutions. Result sets are stored in tables (often by downstreamed ETL processes), others provide static SQL views. The public health insurance organizations of Austria use a self-implemented tool in which simple query chains can be defined. APMN4BI provides the opportunity that business analysts themselves can model the reuse of query results at a conceptual level and, moreover, the concepts of multi-dimensional cubes including enrichments are transferred to target analysis situations (by operator `useAsCube`) such that other navigation operator can use them.

An important requirement was to model and document the steps business analysts perform such that tacit intentions are made visible. Of course, also

common BI tools provide OLAP operations but they offer less opportunities to record and model users' intentions of an operator invocation. It is not visible which multi-dimensional query was the source of a drill down. In most cases, only the last query result is presented. There are no means that present the difference between two queries at a glance. With APMN4BI, a business analyst can model and document her or his intentions by navigation steps and the semantic difference between two analysis situations becomes visible at a glance.

Analysis situations linked by navigation operators represent analysis processes that can be additionally controlled by navigation guards that also can be regarded as additional semantics of navigation steps. On one side, navigation guards can examine result sets of query executions, on the other side, navigation guards can also be used to examine properties of the source analysis situation. The LEICON and KOTI Kobra case studies contain both variants of navigation guards.

The concept of subgraph provides means of decomposition to keep track of an analysis process. Often, a user wants to look simultaneously at several dependent analysis situations which can be considered as an overall analysis situation. For this requirement, APMN4BI provides the conceptual construct of composite analysis situations. Especially, composite analysis situations were included in the final case studies of LEICON and KOTI Kobra.

7.3.2 Criteria 2: Coherent Language Design

APMN4BI pursues the idea of distinction between schema and instance level. The change from a frame-based representation [30] as emphasized in previous work [91] to a proactive modeling approach as proposed in [92] led to a clear separation of schema and instance level.²⁶ Model constructs of APMN4BI

²⁶Whereas in this thesis there is a clear separation between BI analysis graph schemas, BI analysis graphs, and an analysis trace (also including backtracking), in [91] the notion "BI analysis graph" was to the fore. In [91], a distinction between generic analysis situations (correspond to analysis situation schemas in APMN4BI) and individual analysis situations (correspond to analysis situations in APMN4BI), and between generic navigation steps (correspond to navigation step schemas in APMN4BI) and individual navigation steps (correspond to navigation steps in APMN4BI) was already present but there was no clear

are used for proactive modeling at schema level. At execution time (corresponding to an application of an APMN4BI model) these schema items are instantiated.

The differentiation of schema and instance level already starts at the underlying eDFM. Instances of cubes are described by cube schemas. A cube schema comprises dimension schemas. Cube schemas are used in analysis situation schemas. Analysis situation schemas describe multi-dimensional queries and can be instantiated to obtain analysis situations that correspond to specific multi-dimensional queries which are applied to cubes (i.e., to instances of the corresponding cube schemas) comprising dimensions (i.e., instances of the corresponding dimension schemas).

Analysis situation schemas represent source and target of navigation step schemas. Navigation step schemas are instantiated to obtain specific navigation steps. Navigation guards are constituents of navigation step schemas as well as of instances of navigation step schemas (i.e., of navigation steps). BI analysis graph schemas describe BI analysis processes whereas a BI analysis graph represents an instance of a BI analysis graph schema and represents the execution of a specific BI analysis process. Formally, BI analysis graph schemas are represented by directed multi-graphs whereas BI analysis graphs are formally defined as directed trees. Also subgraphs are considered at schema and at instance level. Especially, composite analysis situation schemas belong to BI analysis graph schemas and composite analysis situations are subgraphs of BI analysis graphs.

All model constructs are defined in a mathematical formalism at schema and instance level, and have graphical representations that also allow a distinction between schema and instance level. The visualization of analysis sit-

distinction between BI analysis graph schemas and BI analysis graphs (i.e., instances of BI analysis graph schemas). In fact, it was a view of a cohesive development (design) and usage process. A BI analysis graph was contemporary involved in design and use steps. A BI analysis was considered as “an alternate sequence of individual analysis situations and navigation steps that represent a sequential trace of the analysis steps performed by an analyst in a particular analysis” (somehow, a mixture of BI analysis graphs and analysis traces in APMN4BI). The notions of analysis situation schemas, navigation step schemas and, particularly, BI analysis graph schemas, as well as the notion of “proactive modeling” just arose in [92]. In spite of it all, exact formal definitions which distinguish between schema and instance level were yielded during the final elaboration of this thesis.

uations indicates affiliation to schema or instance level by specific symbolic: double-edged borders for analysis situation schemas; single-edged borders for instances of analysis situation schemas. Thus, BI analysis graph schemas comprise analysis situation schemas that are decorated by double-edged borders and BI analysis graphs are identifiable by analysis situations with single-edged borders. Also for the graphical representation of the condensed form of subgraphs and, especially, for visualization of composite analysis situation schemas and composite analysis situations, double-edged and single-edged borders are used for distinction between schema and instance level.

With respect to the symbolic in graphical representations of APMN4BI, a coherent language design was targeted and elaborated over the years. For instance, one of the last adaptations were made when also using pictograms of an eDFM for constituents of analysis situations and for actual parameters of navigation operators. It was important that a user can easily associate similar things by similar symbols. Or, as another example, the use of non-comparative navigation operators in the application of comparative navigation operators is justified by the argument that there exists a relation between contexts of comparative analysis situations or between contexts of comparative analysis situations and non-comparative analysis situations.

Although the evaluation of this design criteria is mainly based on informed arguments and static analysis, the elaboration of the coherent language design was also strongly driven by insight gained from the case studies. Thus, for instance, in earlier versions of APMN4BI, there was more blending of schema and instance constructs. Only in the course of time, a clear distinct between schema and instance level was found formally and visually. Despite of this separation, also an iterative and evolutionary development process for analysis processes can be performed by switching between schema and instance level in a steadily manner.

7.3.3 Criteria 3: Completeness of Model Constructs

As described in 7.3.1, elaboration and evaluation of APMN4BI was accomplished in discussions and interviews with business analysts and subject mat-

ter experts in the context of real use cases. In these case studies, we also turned our attention to completeness of model constructs.

It was important to cover all activities necessary for OLAP analysis. Thus, common multi-dimensional data models represented conceptually by eDFM's were required. Queries based on such data models can be represented by non-comparative analysis situations. Their constituents comprise all elements of a multi-dimensional query at a conceptual level (cube, aggregate measures, base measure conditions, filter conditions, and dimension qualifications containing dice levels, dice nodes, slice conditions, and granularity levels). Elementary OLAP operations can be performed by non-comparative navigation operators: moving from one to another dice node, changing granularity levels, changing aggregate measures, changing slice conditions, base measure conditions, and aggregate measure filters.

Furthermore, the semantics of common multi-dimensional data models and OLAP operations was enriched and refined, and, over the years based on use cases and case studies, more and more new elements were incorporated in the eDFM. The definition of base measures and aggregate measures were refined including derived base measures and derived aggregate measures. Dimensional predicates, dimensional operators, base measure predicates, and aggregate measure predicates, all including optional subsumption hierarchies, become elements of an eDFM and can be used for non-comparative analysis situations to define OLAP queries.

Comparison, as another new concept, was introduced conceptually which, in general, is beyond the scope of common OLAP queries but which proved to be an important activity of business analysts. This type of comparison takes into account two non-comparative analysis situations and already appeared in the early stage of APMN4BI (see [91]). Also comparative navigation steps (e.g., realized by operator `correlate`) and the usage of scores, all based on real use cases, were already presented in a first version in [91]. After further use cases and case studies, elements used in comparative analysis situations were incorporated in an eDFM. Thus score definitions and, based on the insights of final cases studies, also score predicates and join conditions become elements of an eDFM. A comparative analysis situation and a comparative

analysis situation schema comprise two non-comparative analysis situations or two non-comparative analysis situation schemas, respectively, and join conditions, scores, and score predicates.

The repertoire of navigation operators (non-comparative and comparative) was also successively elaborated and evaluated over the years by real use cases. On the one side, it was important to change all constituents of analysis situations, on the other side, some navigation operators also represent higher level constructs. For instance, some operators can be used to iterate over a sequence of dimension nodes (e.g., by operators `moveToFirstNode` and `moveToNextNode`, other operators are used to express additional semantics, for example, operator `narrowSliceCond+` narrows a slice condition by adding an additional dimensional predicate and operator `narrowSliceCond->` narrows a slice condition by exchanging a dimensional predicate by a stronger one. Also comparative navigation operators that, in most case, additionally use non-comparative operators, and also the `useAsCube` operator were carefully elaborated and evaluated by real use case. Finally, navigation steps are controlled by navigation guards. The definition of navigation steps and navigation step schemas, and all navigation operators of Chapter 4 can be considered a result of use cases and case studies.

The definition of BI analysis graph schemas is closely linked to the usage of variables. Variables are used for constituents of analysis situation schemas and for actual parameters in navigation step schemas. At different stages in the development of APMN4BI, different intensity of variable usage was proposed. During the final case studies, the insight arose to allow variables for every constituent of an analysis situation schema, even for cubes. Only one restriction was claimed indicating that the underlying cube schema of an analysis situation schema is fixed. Also the generic definition of navigation step schemas allow that every actual parameter may be a variable. The possibility, to allow variables almost anywhere, provides the possibility to maximize generalization in BI analysis graphs schemas.

Although composite analysis situations were introduced in an early stage of APMN4BI (see [91]), the need to structure BI analysis graph schemas and BI analysis graphs was formally defined later. This need was emphasized in

the final case studies. The definition of BI analysis graph schemas as directed multi-graphs and the definition of BI analysis graphs as directed trees provided formally an easy way for a hierarchical structuring by subgraphs. As a consequence, composite analysis situation schemas and composite analysis situations also represent subgraphs that have additional specific features as described in Section 6.5.

Together with the argumentation of 7.3.1 and the fact of long-term elaboration and evaluation (since the project start of semCockpit in 2011 [94, 91]) in co-operation with customers we argue that APMN4BI provides a complete set of constructs for modeling and documenting BI analysis processes at a conceptual level. Additional insights were gained by prototype implementations in university courses (see Section 1.6) and, especially, by the real environments of the case studies as presented in Section 1.3.

7.3.4 Criteria 4: Mapping to SQL

One aim of APMN4BI was to provide realizable conceptual constructs that can be implemented in modeling and execution tools. In this sense, we claimed that analysis situations are mapped into SQL and navigation steps that link two analysis situations yield precise semantics with respect to the translation into SQL (design criteria 4). By static analysis and informed arguments, we provide an evaluation of the fulfillment of this design criteria.

Analysis situations represent queries based on multi-dimensional cubes. We propagated an approach that implements multi-dimensional queries on the basis of relational database tables where eDFM's are provided as star schemas (see [62]). We presuppose simple dimensions with only one hierarchy which are linked to fact tables. Equal key names are chosen for both the primary key of a dimension and the corresponding foreign key in the fact table. Hence, SQL joins can be simplified by natural joins. Moreover, the whole eDFM assumes unique name for all items such that name clashes and aliasing can be avoided. Furthermore, we expect that enrichments of the eDFM (dimensional operators, dimensional predicates, base measures, aggregate measures, base measure predicates, aggregate measure predicates,

join conditions, scores, and score predicates) are constructs that are represented by correct SQL expressions. The translation of an eDFM into a star schema is presented and formally defined in Section 2.4.4.

Non-comparative analysis situations can be translated into SQL statements as shown in Section 3.1.4. All conceptual constructs of a non-comparative analysis situation have a unique position in the resulting SQL statement. Granularity levels and aggregate measures represent the projection, base measure filters, dice levels and dice nodes, and slice conditions represent parts of the where clause, filter conditions are mapped to the having clause, and cube (i.e., the fact table) and dimensions are joined by natural joins. Aggregate measure names serve as alias names. Slice conditions, filter conditions, base measure conditions, and aggregate measures are recursively replaced by their underlying expressions.

Analogously, comparative analysis situations can be translated into SQL statements as shown in 3.2.4. The context of interest and the context of comparison are translated in SQL statements accordingly to non-comparative analysis situations. Both are joined by the join conditions and both are referenced by alias names *CoI* and *CoC*. Scores are additionally added to the projection, and score filters are retrieved as having clauses in the outer query.

In Section 4.4 derived cubes were introduced. A non-comparative analysis situation is used to define a derived cube via navigation operator *useAsCube*. The target analysis situation of such a navigation step already uses the derived cube. Section 4.4.3 demonstrates the translation of derived cubes into SQL. This translation is realized by views which are based on the query of the non-comparative source analysis situation of the *useAsCube* operation.

Finally, navigation steps can be considered as query transformations. The query corresponding to the source analysis situation is transformed into a query corresponding to the target analysis situation. In this sense, a navigation step can also be considered as a transformation of SQL statements. The SQL statement extracted from the source analysis situation is transformed into an SQL statement obtained from the target analysis situation.

7.3.5 Criteria 5: Early consistency checking

To avoid or mitigate modeling and execution errors, APMN4BI was designed to allow early consistency checking. In Section 5.2, we introduced the notion of type-compliant navigation steps in the sense of navigation steps that are compliant with respect to a navigation step schema (also called schema-compliant) which means that if a navigation operator of a navigation step schema is applied to an instance of the source analysis situation schema of that navigation step schema, then the resulting analysis situation has to be an instance of the target analysis situation schema.

In Subsection 5.2.3 of Chapter 5, type compliance is discussed in detail and, in this context, also the notion of type safety is used. To check type safety, it is necessary to check whether the pre- and postcondition of a navigation operator (including the frame assumption) are satisfied or not. Preferably, such a type checking should be performed at the earliest possible moment. If type safety can already be examined at schema level, errors can already be avoided at modeling time. On the other side, it is necessary to check type safety at least at runtime (i.e., at instantiation time or at the latest before the query of an analysis situation is executed). In this sense, analogously to programming languages, we distinguish between static and dynamic type safety. Checking static type safety (static type checking) represents the checking of type safety at schema level whereas checking dynamic type safety (dynamic type checking) can be considered as checking of type safety at instance level.

Whether already static or only dynamic type checking is possible depends on the constellation of constants and variables in a navigation step schema. Table 5.2 presented in Subsection 5.2.3 gives an overview of all relevant configurations of constants and variables that can occur in a navigation step schema. Depending on these configurations, this table comprises information whether static or only dynamic type checking is possible. The table shows that if all properties in the source analysis situation schema and all actual parameters of the navigation operator are constant, and if at most only the target analysis situation schema contains variables, then static type

checking can be performed. In all other cases, only dynamic type checking is possible. Two configurations of variables and constants are not reasonable. In this way, for example, for implementation of modeling tools and runtime environments for APMN4BI, Table 5.2.3 can be used to decide whether type checking can be performed during the modeling phase (static type checking) or at runtime (dynamic type checking). Additionally, navigation guards provide further means to model restrictions for controlling navigation at runtime explicitly.

7.3.6 Criteria 6: Reasonable Decisions Regarding Trade-offs

To simplify the definition and usage of APMN4BI models, we made some trade-offs that provide more advantages than disadvantages: (1) We only allow one hierarchy per dimension. (2) We do not allow dimension roles. (3) We introduced navigation guards instead of inheritance.

The first two trade-offs regard the eDFM. As mentioned in Section 2.4.1, we only allow one hierarchy per dimension (as a difference to [34]).²⁷ We made this design decision to simplify the definition and usage of APMN4BI. If we would allow more than one hierarchy per dimension, the user also would have to additionally indicate hierarchies in dimension qualifications and navigation operator parameters. Hierarchies themselves would have to have names. In our approach each hierarchy corresponds to exactly one dimension, i.e., the hierarchy name would also have to be encoded in the dimension name.

Although, we have to manage a higher amount of dimensions (as a consequence of this design decision that allows to have only one hierarchy per dimension), they (dimensions with only one hierarchy) can be used in a sim-

²⁷In [91] and also in [92], it was allowed to have dimensions with more than one hierarchy. Furthermore, it was allowed to have parallel as well as alternative hierarchies in one dimension. Because, in this thesis, we only allow one hierarchy per dimension, all hierarchies (dimensions, respectively) have to be considered as parallel dimension hierarchies. The correct use of dimension hierarchies (dimensions, respectively) is the responsibility of the user (e.g., the business analyst).

pler way. There exists only one name per dimension that correspond only to one hierarchy. Otherwise, referring a hierarchy, one always would have to state both the name of a dimension and the name of a hierarchy. Concerning implementation, a dimension in an eDFM can be realized by one database table and this database table comprises exactly one dimension hierarchy.

The second trade-off concerns dimension roles as briefly discussed in Section 2.4.3. We do not allow dimension roles, i.e., one dimension cannot be used twice for one cube (as a difference to [34]). This decision was made to keep the formal definitions of cube schemas and cube instances as simple as possible. In addition to it, the translation into SQL was not complicated, too. Furthermore, additional qualification would be necessary when referring to dimension levels, descriptive attributes, or dimensional predicates. Nevertheless, it was recognized in case studies that dimension roles would enhance the usage of APMN4BI in the sense of better re-usability. Thus, the decision against dimension roles is rather caused in easiness than usefulness.

With respect to a relational implementation of an eDFM, dimension roles can be simulated by additional SQL views on the actual dimension table. This means that a dimension can be realized by one dimension table that is filled with data by one ETL process. For each dimension role, an SQL view can be defined based on this dimension table. Each dimension role can be used in APMN4BI like a common dimension, although, it is efficiently implemented in a relational database. Moreover, dimension roles also could be visualized in an eDFM, for instance, by drawing a dimension schema only once and by connecting such a dimension schema more than once to the “cube rectangle” in such a way that each connection obtains a unique name. As a consequence this name also has to serve as a name space, i.e., dimension levels, descriptive attributes, and dimensional predicates have to be qualified by the name of the dimension role. In this sense, one could also introduce dimension roles at a conceptual level.

Similar to dimension roles that facilitate the re-usability of dimensions, it would also be worth considering entity types for dimension levels. In [91], entity classes (entity types) were proposed. Entity classes describe dimension levels having, optionally, descriptive attributes as additional properties. An

entity type (entity class) helps to increase re-usability. Dimension levels of different dimensions with equal properties can be ascribed to one entity type. Such an entity type determines the characteristics of one dimension level.

The third trade-off concerns the option to generalize or specialize analysis situation schemas by inheritance. The discussions with users and the experience gathered in real use cases showed that business analysts prefer a conditional approach instead of inheritance. Thus, we use navigation guards instead of inheritance to differentiate between specialized analysis situations. The loss of additional opportunities for type checking via “base analysis situation schemas” (corresponding to base classes in the sense of an object-oriented design) is less unfavorable than the risk of less comprehensibility and less user acceptance.

Chapter 8

Conclusion and Possible Extensions

In this thesis, a conceptual graphical modeling language for business intelligence analysis processes based on dimensional fact models was introduced. The notation provides expressiveness that allows to model activities of business analysts in an easy understandable manner. For this reason, we go without existing modeling languages like, for example, UML, BPMN, or IFML. APMN4BI represents a conceptual domain-specific language for modeling BI analysis processes based on OLAP operations. All modeling elements of APMN4BI can be considered as first class citizens. Analysis situations correspond to multi-dimensional queries that are performed by business analysts. Navigation operations document navigation steps of business analysts. Their intentions are made visible in the sense that a navigation operation shows the semantic difference between source and target analysis situation. Comparisons are main activities of business analysts and are represented as comparative analysis situations that relate a context of interest to a context of comparison. The dynamic process view of APMN4BI is based on a static data view represented in an eDFM. Derived cubes can be considered as another construct that introduces an eDFM view on the basis of a non-comparative analysis situation. An eDFM goes beyond a common DFM by introducing further constructs like, for instance, dimensional predicates. A separation be-

tween schema and instance level (across all APMN4BI constructs including eDFM) provides a clear distinction between modeling and execution activities. The control of analysis processes can be modeled by BI analysis graph schemas. Navigation operators, variables, and navigation guards are used to obtain useful paths for data analysis. BI analysis graphs represent instances of BI analysis graph schemas and can be considered as concrete analysis processes. Subgraphs provide means for hierarchically structuring of BI analysis graph schemas and BI analysis graphs. A composite analysis situation as a subgraph provides additional features such that it can be considered as one “analysis situation” that is composed by further analysis situations and navigation steps, and such that all included analysis situations are created and executed in one go.

This thesis also provides a translation of conceptual APMN4BI constructs into SQL. The mapping into SQL was claimed as a separate design criteria of APMN4BI (Section 1.4.2 and Section 7.3.4) and does not only demonstrate a possible way for implementing tool support but, particularly, it provides a basis to obtain precise semantics for APMN4BI. First of all, an eDFM is translated into a relational star schema (Section 2.4.4). Additional constructs of an eDFM (dimensional operators, dimensional predicates, base measures, base measure predicates, aggregate measures, aggregate measure predicates, scores, score predicates, and join conditions) are treated as appropriate SQL expressions. Non-comparative and comparative analysis situations are translated into SQL queries (Section 3.1.4 and Section 3.2.4), and derived cubes can be considered as SQL views which are represented conceptually as additional eDFM’s. Finally, the navigation operators presented in Chapter 4 are used in navigation steps to reveal the semantic difference of two SQL queries (concerning the source and target analysis situation of a navigation step) at a conceptual level.

APMN4BI as a design artifact (see [41]) represents a new language construct for modeling BI analysis processes. The elaboration of APMN4BI was based on real business environments (public health insurance organizations in Austria but also in Germany, pension insurance organization of Austria, and an Austrian company for brush manufacturing that is a subsidiary of a

bigger European company group) that emphasize a certain substantial problem relevance. Use cases and case studies from these business areas flow in the elaboration and evaluation of APMN4BI over many years. Beside static analysis and informed arguments, case studies represent an important basis for design evaluation of APMN4BI. Especially, the final case studies executed in 2018 and presented in [87, 88, 89] led to a final elaboration cycle of APMN4BI that results in the incorporation of additional features. Features that were not incorporated (especially due to reduce complexity in formal definitions and to keep the presentation of APMN4BI as simple as possible), and general advanced concepts and ideas are shortly presented in the subsequent paragraphs in the sense of an “extended APMN4BI”.

One class of additional features towards an extended APMN4BI concerns the eDFM. As discussed in Chapter 7 (especially, Section 7.3.6), it would be worth considering more than one hierarchy in one dimension, and to introduce dimension roles to increase re-usability. To provide these advanced APMN4BI constructs, navigation step schemas and navigation steps have to be appropriately extended for also referring dimension hierarchies and dimension roles. In addition to it, also the translation into SQL has to be adapted.

Additionally to increase encapsulation and re-usability, name spaces and entity types can be introduced in an extended APMN4BI. Name spaces allow to reuse names for dimension levels and descriptive attributes in various dimensions or dimension hierarchies. Such names need not be globally unique any more—they are locally encapsulated. Moreover, dimension levels and descriptive attributes could be encapsulated in entity types and reused in various dimensions or dimension hierarchies. Again, also in these extensions, not only the eDFM itself but also navigation step schemas and navigation steps must be adapted to enable correct qualification of dimension levels and descriptive attributes. Furthermore, name spaces and entity types can be used to comprise dimensional operators and dimensional predicates such that these constructs are once defined and reused in several dimensions or dimension hierarchies.

Another advanced elements concerning eDFM’s represent multi-ary pred-

icates (dimensional predicates, base measure predicates, aggregate measure predicates, and score predicates). Allowing parameters, one can provide more general predicate definitions and more universal usage of such predicates. Moreover, multi-ary predicates can also refer to lookup tables that comprise values which are accessed and used in the predicate definition (see also Section 7.2.3.1).

As defined in Section 2.4.1.1, dimensional predicates are aligned to one dimension schema that can be used as slice conditions contained in a dimension qualification which is also linked to a dimension. Similar to [91] where multi-dimensional concepts are presented, another extension of APMN4BI represents the definition of multi-dimensional predicates that uses dimensional predicates based on multiple dimensions. Moreover, a predicate definition could even use dimensional predicates and base measure predicates to provide a superior predicate. For extending predicate definitions in this sense, also the definition of analysis situations and navigation steps would have to be adapted.

The second category of additional features towards an extended APMN4BI refers to navigation steps (also discussed in Chapter 7). Navigation operators as introduced in Chapter 4 represent elementary operations mainly based on OLAP operations. There are use cases comprising sequences of navigation steps where only the source analysis situation of the first and the target analysis situation of the last navigation step are of main interest. All intermediate analysis situations are only considered as auxiliary ones (“uninteresting”, useless, or, even, meaningless analysis situations). In an extended APMN4BI, sequences of navigation step schemas and navigation steps can be formally introduced as a separate conceptual construct. Furthermore, such an intermediate (“uninteresting”) analysis situation also might occur as a source analysis situation for several sequences of navigation steps where each sequence corresponds to a separate branch. Such a constellation of navigation steps and navigation step schemas represents a fork that, in an extended APMN4BI, can also be considered as an additional construct at a conceptual level.

Another insight gained from case studies concerns further use of the query

result set of an analysis situation. An advanced APMN4BI could allow to define predicates (dimensional predicates, base measure predicates, aggregate measure predicates, and score predicates) that are based on the result set of an analysis situation. There are use cases in which an analysis situation needs, for instance, slice conditions that are dependent on the result set of another preceding analysis situation. This is different to a derived cube which is based on the original cube of the source analysis situation. Hence, the use of a query result set of an analysis situation may concern an analysis situation that is founded on a completely different cube. The use of such a result set is similar to the access to lookup tables as previously mentioned in the context of multi-ary predicates. That means, the result set of an analysis situation can be used in a predicate definition of another eDFM.

In Section 1.4.1, aims for APMN4BI were formulated. As demonstrated in this thesis, APMN4BI represents a precise, unambiguous, and understandable conceptual modeling language for BI analysis processes that makes the intention of business analysts visible when performing an analysis step. On one side, APMN4BI can be used for documenting existing BI analysis processes, on the other side, it can be used for specifying new BI analysis processes.

In the first case (documentation of BI analysis processes), many analysis processes in real business environments were found that are worth to be documented, especially to increase comprehensibility, traceability, replicability, and transparency. This aim comes in the fore more and more with respect to data privacy in recent years. Especially, the introduction of the General Data Protection Regulation (GDPR) by the European Union in 2018 presents companies and public organizations with new challenges. In particular, Austria's public health insurance organizations and Austria's public pension insurance PVA are faced with new requirements concerning data privacy, especially also for BI and data warehouse systems. For instance, each access to private data has to be recorded and also justified. In this context, APMN4BI (combined with appropriate tool support) can become more important. APMN4BI was designed to provide a language for controlling and documenting such analysis processes, and, thus, is predestined to be

applied for supporting the fulfillment of such requirements. Together with a presentation & action layer (compare Section 2.2), justification information can be stored and audit actions can be invoked. Another example represents suppressing of result records (yielding from aggregation) that refer to a small amount of human beings. Usually, anonymous aggregated data can be considered as uncritical with respect to data privacy except it is visible that an aggregation comprises only few people (in this context, the number “three” is often considered as a critical amount of human beings). In this case, individuals could be investigated in combination with other features like, for instance, sex, age, and appropriate details about residence. Again, APMN4BI could be used to define analysis processes that filter such records and that provides together with the presentation & action layer appropriate documentation, justification, and optional actions.

For specifying BI analysis processes, APMN4BI is used at schema level proactively to model analysis processes. The modeling process is performed in an iterative and evolutionary manner (models are specified, instantiated, executed, and, after evaluation, extended, adapted, and refined). As previously mentioned, the provisions concerning data privacy become more important in recent years. Austria’s public health insurance organizations and Austria’s public pension insurance PVA started to define “technical and organizational measures (TOM’s)” to meet the requirements of a GDPR-compliant BI and data warehouse system. The specification of GDPR-compliant BI analysis processes can be supported by modeling such processes by APMN4BI.

Moreover, APMN4BI was designed to facilitate the reuse of BI analysis processes (another aim of APMN4BI). The modeling approach at schema level represented by BI analysis graph schemas including the use of variables and navigation guards allows to provide specifications of generalized analysis processes that represent a broad class of several concrete analysis processes. Finally, the specification of BI analysis processes by APMN4BI models provides the basis for implementing such processes (aim 6 in Section 1.4.1) and, furthermore, APMN4BI was designed to define the basis for implementing modeling tools for APMN4BI itself and for tools to automate BI analysis process execution based on APMN4BI models (aim 7 in Section 1.4.1).

To fulfill the aims of APMN4BI presented in Section 1.4.1, design criteria were defined in Section 1.4.2 and evaluated in Section 7.3. Accordingly to these design criteria, APMN4BI represents a domain specific conceptual modeling language that respects a coherent language design concerning the distinction between schema and instance level, and concerning visualization. The completeness of model constructs were elaborated and evaluated by use cases and case studies from real business environments over about ten years. The mapping into SQL provides precise semantics. Early consistence checking as a separate design criteria facilitates the development and use of tools for specifying APMN4BI models and for executing such models. Decisions for trade-offs were justified in Section 7.3.6. Together with the fulfillment of the design criteria presented in Section 1.4.2, APMN4BI can be considered as a precise, unambiguous, and understandable conceptual modeling language for BI analysis processes that makes the intention of a business analyst visible when performing analysis steps, that can be used for documenting and specifying BI analysis processes such that reuse is facilitated, and such that a basis for implementing BI analysis processes and the basis for implementing tool support is provided.

List of Figures

2.1	Health care use case – overview of an analysis process	55
2.2	Model layers for APMN4BI	58
2.3	eDFM of the running example	107
2.4	Abstract visualization of an example of a cube instance	108
2.5	Example of a translation of a cube instance into a star schema	109
3.1	Template of a non-comparative analysis situation depicted in full graphical representation	118
3.2	Example of a non-comparative analysis situation depicted in full graphical representation	119
3.3	Template of a non-comparative analysis situation depicted in lean graphical representation	120
3.4	Example of a non-comparative analysis situation depicted in lean graphical representation	121
3.5	Template of a non-comparative analysis situation depicted in condensed graphical representation	121
3.6	Example of a non-comparative analysis situation depicted in condensed graphical representation	122
3.7	Formal specification of the translation of a non-comparative analysis situation into SQL	124
3.8	Relational schema of a result set of a non-comparative analysis situation	125
3.9	Example of the translation of a non-comparative analysis sit- uation into SQL	125
3.10	Example of a result set of a non-comparative analysis situation	126

3.11	Template of a comparative analysis situation depicted in full graphical representation	134
3.12	Example of a comparative analysis situation depicted in full graphical representation	137
3.13	Template of a comparative analysis situation depicted in lean graphical representation	138
3.14	Example of a comparative analysis situation depicted in lean graphical representation	138
3.15	Template of a comparative analysis situation depicted in condensed graphical representation	139
3.16	Example of a comparative analysis situation depicted in condensed graphical representation	139
3.17	Formal specification of the translation of a comparative analysis situation into SQL	141
3.18	Relational schema of a result set of a comparative analysis situation	143
3.19	Example of the translation of a comparative analysis situation into SQL	146
3.20	Example of a result set of a comparative analysis situation	147
4.1	Navigation operator examples	160
4.2	Example of a comparative navigation step containing first variant of operator relate	185
4.3	Example of a comparative navigation step containing second variant of operator relate	186
4.4	Example of a comparative navigation step containing operator target	188
4.5	Example of a comparative navigation step containing a variant of operator rerelate	194
4.6	Example of three comparative navigation steps containing operators correlate and narrowScoreFilter+	195
4.7	Example of a comparative navigation step containing operator unrelate	200

4.8	Example of a navigation step with operator <code>useAsCube</code>	211
4.9	SQL view of the derived cube of the non-comparative source analysis situation of the example in Figure 4.8	212
4.10	SQL statement of the target analysis situation of the example in Figure 4.8	212
4.11	eDFM of the example in Figure 4.8	216
4.12	Example of a comparative navigation step containing a navigation guard	217
5.1	Non-comparative analysis situation schema with unbound variables (full graphical representation)	225
5.2	Non-comparative analysis situation schema with unbound variables (lean graphical representation)	226
5.3	Non-comparative analysis situation schema with unbound variables (condensed graphical representation)	226
5.4	Non-comparative analysis situation schema without variables (lean graphical representation)	227
5.5	Non-comparative analysis situation schema without constants (lean graphical representation)	227
5.6	Non-comparative analysis situation schema with named variables (lean graphical representation)	228
5.7	Comparative analysis situation schema with unbound variables (full graphical representation)	231
5.8	Comparative analysis situation schema with unbound variables (lean graphical notation)	232
5.9	Navigation step schemas with unbound variables	251
5.10	Navigation step schemas with constant properties in the source analysis situation schema and type checking options	257
5.11	Navigation step schemas with a variable property in the source analysis situation schema and type checking options	258
5.12	Navigation step schema with unbound variables involving a comparative analysis situation	260
5.13	Navigation step schemas with equal source and target	263

5.14	Example with navigation guard using operator <code>diceNode</code> . . .	264
5.15	Example with navigation guard using operator <code>hasResult</code> . .	267
5.16	Navigation pattern to drill down to the next finer granularity level	267
5.17	Instances generated by navigation pattern of Figure 5.16 . . .	268
5.18	Navigation pattern of Figure 5.16 extended by a navigation guard to check the navigation operator's precondition	269
5.19	Navigation pattern to drill down to the next finer granularity level depending on the result set of the source	270
5.20	Instances generated by navigation pattern of Figure 5.19 . . .	270
5.21	Navigation pattern to iterate over subnodes of a dimension node	271
5.22	Navigation pattern to iterate over subnodes of a dimension node depending on the result set of the source	272
5.23	Navigation pattern iterating over subnodes and drilling down to sublevel depending on a navigation guard	278
5.24	Example of Figure 5.23 presenting analysis situation schemas in condensed graphical notation	279
6.1	Example of a BI analysis graph extracted from the running example presented in Section 2.1	288
6.2	Example of a BI analysis graph schema extracted from the running example	295
6.3	Modified BI analysis graph schema of Figure 6.2 containing more than one navigation step schema between two analysis situation schemas	298
6.4	Example of a BI analysis graph schema comprising navigation step schemas having same analysis situation schema as source and target	299
6.5	Example of a BI analysis graph schema having two navigation step schemas where the target analysis situation schema of one navigation step schema becomes the source of the other navigation step schema, and vice versa.	301

6.6	A smaller instance of BI analysis graph graph schema of Figure 6.1	307
6.7	An instance of a BI analysis graph graph schema of Figure 6.1 consisting of only one vertex	308
6.8	Instance of BI analysis graph schema of Figure 6.4	310
6.9	Hierarchical structuring of a BI analysis graph schema by subgraphs (corresponding to the example of Figure 6.2)	312
6.10	Visual decomposition of BI analysis graph schema of Figure 6.2 into subgraphs (corresponding to the tree representation in Figure 6.9)	313
6.11	Hierarchical structuring of a BI analysis graph by subgraphs (corresponding to the example of Figure 6.1)	315
6.12	Visual decomposition of BI analysis graph of Figure 6.1 into subgraphs (corresponding to the tree representation in Figure 6.11)	316
6.13	Embedded subgraphs <i>AG-1</i> , <i>AG-2</i> , and <i>AG-3</i> at schema level .	318
6.14	Embedded subgraphs <i>ag-1</i> , <i>ag-2</i> , and <i>ag-3</i> at instance level . .	319
6.15	Embedded subgraphs <i>AG-2.1</i> and <i>AG-2.2</i> at schema level . . .	320
6.16	Boundary between subgraphs <i>AG-2.1</i> and <i>AG-2.2</i>	321
6.17	Variants <i>AG-B.2.1</i> and <i>AG-B.2.2</i> of embedded subgraphs at schema level	323
6.18	Embedded subgraphs <i>ag-2.1</i> , <i>ag-2.2.1</i> , and <i>ag-2.2.2</i> at instance level	324
6.19	Embedded subgraphs <i>ag-B.2.1</i> , <i>ag-B.2.2.1</i> , and <i>ag-B.2.2.2</i> at instance level	325
6.20	Running example presented in Figure 6.2 including a composite analysis situation schema	339
6.21	Example of Figure 6.20 in condensed graphical representation	341
6.22	Composite analysis situations instantiated from composite analysis situation schema presented in Figure 6.20	342
6.23	The same example as presented in Figure 6.20 except another assemblies of analysis situation schemas to composite analysis situation schemas	344

6.24	Composite analysis situations instantiated from composite analysis situation schemas presented in Figure 6.23	346
6.25	Alternative modeling of the composite analysis situation schemas presented in Figure 6.23	347
6.26	Example of a composite analysis situation schema containing a loop	349
6.27	A composite analysis situation instantiated from composite analysis situation schema presented in Figure 6.26	351

List of Tables

1.1	Claimed design criteria and evaluation strategies	25
4.1	Operators used in navigation guard expressions of a non-comparative analysis situation as or comparative analysis situation cas . We use an object-oriented style to indicate the application of such an operator to as and cas	156
4.2	Operators <code>drillDownOneLevel</code> , <code>drillDownToLevel</code> , <code>rollUpOneLevel</code> , and <code>rollUpToLevel</code>	159
4.3	Operators <code>moveDownToNode</code> , <code>moveUpToNode</code> , <code>moveAsideToNode</code> , and <code>moveToNode</code>	163
4.4	Operators <code>moveDownToFirstNode</code> and <code>moveDownToLastNode</code> .	164
4.5	Operators <code>moveToNextNode</code> and <code>moveToPrevNode</code>	165
4.6	Operators <code>narrowSliceCond+</code> , <code>narrowSliceCond-></code> , <code>broadenSliceCond-</code> , <code>broadenSliceCond-></code> , <code>refocusSliceCond</code> , and <code>refocusSliceCond-></code>	167
4.7	Operators <code>narrowBMsCond+</code> , <code>narrowBMsCond-></code> , <code>broadenBMsCond-</code> , <code>broadenBMsCond-></code> , <code>refocusBMsCond</code> , and <code>refocusBMsCond-></code>	171
4.8	Operators <code>addMeasure</code> , <code>dropMeasure</code> , <code>refocusMeasure</code> , <code>refocusMeasure-></code> , <code>moveDownToMeasure</code> , and <code>moveUpToMeasure</code> .	174
4.9	Operator <code>narrowFilter+</code> , <code>narrowFilter-></code> , <code>broadenFilter-</code> , <code>broadenFilter-></code> , <code>refocusFilter</code> , and <code>refocusFilter-></code> . .	177
4.10	Operator <code>drillAcrossToCube</code>	179
4.11	Operator <code>relate</code>	183
4.12	Operator <code>target</code>	187

4.13	Operator <code>rerelate</code>	190
4.14	Operator <code>retarget</code>	191
4.15	Operator <code>correlate</code> (part 1)	192
4.16	Operator <code>correlate</code> (part 2)	193
4.17	Operators <code>narrowScoreFilter+</code> , <code>narrowScoreFilter-></code> , <code>broadenScoreFilter-</code> , and <code>broadenScoreFilter-></code>	197
4.18	Operators <code>refocusScoreFilter</code> and <code>refocusScoreFilter-></code>	198
4.19	Operator <code>rejoin</code>	199
4.20	Operators <code>unrelate</code> and <code>untarget</code>	199
4.21	Operator <code>useAsCube</code>	208
5.1	Operators used in navigation guard expressions and applied to a non-comparative analysis situation as or to a comparative analysis situation cas	243
5.2	Type checking options depending on the configuration of constants and variables as properties of the source analysis situation schema, as actual parameters, and as properties of the target analysis situation schema.	248

Bibliography

- [1] Z. E. Akkaoui, E. Zimányi, J. Mazón, and J. Trujillo. A BPMN-based design and maintenance framework for ETL processes. IJDWM, 9(3):46–72, 2013.
- [2] J. Aligon, K. Boulil, P. Marcel, and V. Peralta. A holistic approach to OLAP sessions composition: The falso experience. In Proceedings of the 17th International Workshop on Data Warehousing and OLAP, DOLAP 2014, Shanghai, China, November 3-7, 2014, pages 37–46, 2014.
- [3] J. Aligon, E. Gallinucci, M. Golfarelli, P. Marcel, and S. Rizzi. A collaborative filtering approach for recommending OLAP sessions. Decision Support Systems, 69:20–30, 2015.
- [4] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia. Similarity measures for OLAP sessions. Knowledge and Information Systems, 39(2):463–489, 2014.
- [5] S. Anderlik, B. Neumayr, and M. Schrefl. Using domain ontologies as semantic dimensions in data warehouses. In Conceptual Modeling - 31st International Conference ER 2012, Florence, Italy, October 15-18, 2012. Proceedings, pages 88–101, 2012.
- [6] M. Aufaure, N. Kuchmann-Beauger, P. Marcel, S. Rizzi, and Y. Vanrompay. Predicting your next OLAP query based on recent analytical sessions. In Data Warehousing and Knowledge Discovery -

- 15th International Conference, DaWaK 2013, Prague, Czech Republic, August 26-29, 2013. Proceedings, pages 134–145, 2013.
- [7] A. Azoulay. Business intelligence: Guided analytics v. self-service analytics. QGate Blog, <http://www.qgate.co.uk/blog/business-intelligence-guided-analytics-v-self-service-analytics>, 2015.
- [8] D. Barone, L. Jiang, D. Amyot, and J. Mylopoulos. Composite indicators for business intelligence. In M. Jeusfeld, L. Delcambre, and T.-W. Ling, editors, Conceptual Modeling – ER 2011: 30th International Conference, ER 2011, Brussels, Belgium, October 31 - November 3, 2011. Proceedings, pages 448–458, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [9] D. Barone, T. Topaloglou, and J. Mylopoulos. Business intelligence modeling in action: A hospital case study. In J. Ralyté, X. Franch, S. Brinkkemper, and S. Wrycza, editors, Advanced Information Systems Engineering: 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings, pages 502–517, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [10] D. Barone, E. Yu, J. Won, L. Jiang, and J. Mylopoulos. Enterprise Modeling for Business Intelligence. In W. Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, C. Szyperski, P. Bommel, S. Hoppenbrouwers, S. Overbeek, E. Proper, and J. Barjis, editors, The Practice of Enterprise Modeling, volume 68 of Lecture Notes in Business Information Processing, chapter 3, pages 31–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [11] L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, and D. Laurent. A personalization framework for OLAP queries. In DOLAP, pages 9–18, 2005.
- [12] F. Bentayeb and C. Favre. RoK: Roll-up with the K-means clustering method for recommending OLAP queries. In DEXA, pages 501–515, 2009.

- [13] A. Borgida, J. Mylopoulos, and R. Reiter. ”...and nothing else changes”: The frame problem in procedure specifications. In Proceedings of the 15th International Conference on Software Engineering, Baltimore, Maryland, USA, May 17-21, 1993., pages 303–314, 1993.
- [14] A. Borgida, J. Mylopoulos, and R. Reiter. On the frame problem in procedure specifications. IEEE Transaction on Software Engineering, 21(10):785–798, 1995.
- [15] Bundesministerium für Gesundheit und Frauen (BMGF). Gesundheitsziele Österreich. Richtungsweisende Vorschläge für ein gesünderes Österreich — Langfassung. <https://gesundheitsziele-oesterreich.at/10-ziele/>, 2017.
- [16] L. Cabibbo and R. Torlone. From a procedural to a visual query language for OLAP. In 10th International Conference on Scientific and Statistical Database Management, Proceedings, Capri, Italy, July 1-3, 1998, pages 74–83, 1998.
- [17] S. Ceri, M. Brambilla, and P. Fraternali. The history of WebML lessons learned from 10 years of model-driven development of web applications. In Conceptual Modeling: Foundations and Applications, volume 5600 of Lecture Notes in Computer Science, pages 273–292. Springer, 2009.
- [18] S. Ceri, F. Daniel, and M. Matera. Extending WebML for modeling multi-channel context-aware web applications. In 4th International Conference on Web Information Systems Engineering Workshops, WISE 2003 Workshops, Rome, Italy, December 13, 2003, pages 225–233, 2003.
- [19] S. Ceri, P. Fraternali, and A. Bongio. Web modeling language (WebML): a modeling language for designing web sites. Computer Networks, 33(1-6):137–157, 2000.
- [20] S. Ceri, P. Fraternali, and S. Paraboschi. Design principles for data-intensive web sites. SIGMOD Rec., 28(1):84–89, 1999.

- [21] H. Chen, R. H. L. Chiang, and V. C. Storey. Business intelligence and analytics: From big data to big impact. MIS Quarterly, 36(4):1165–1188, Dec. 2012.
- [22] P. P.-S. Chen. The entity-relationship model—toward a unified view of data. ACM Trans. Database Syst., 1(1):9–36, Mar. 1976.
- [23] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User-Analysis: An IT Mandate, 1993.
- [24] T. H. Davenport and D. J. Patil. Data Scientist: The Sexiest Job Of the 21st Century. Harvard Business Review, 90(10):70–76, 2012.
- [25] I. Davies, P. Green, M. Rosemann, M. Indulska, and S. Gallo. How do practitioners use conceptual modeling in practice? Data & Knowledge Engineering, 58(3):358–380, Sept. 2006.
- [26] B. A. Devlin and P. T. Murphy. An architecture for a business and information system. IBM Syst. J., 27(1):60–80, Jan. 1988.
- [27] Z. El Akkaoui and E. Zimanyi. Defining ETL workflows using BPMN and BPEL. In Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP, DOLAP '09, pages 41–48. ACM, 2009.
- [28] D. W. Embley and S. W. Liddle. Big data - conceptual modeling to the rescue. In Conceptual Modeling - 32th International Conference, ER 2013, Hong-Kong, China, November 11-13, 2013. Proceedings, pages 1–8, 2013.
- [29] F. Figge, T. Hahn, S. Schaltegger, and M. Wagner. The sustainability balanced scorecard—linking sustainability management to business strategy. Business strategy and the Environment, 11(5):269–284, 2002.
- [30] R. Fikes and T. Kehler. The role of frame-based representation in reasoning. Commun. ACM, 28(9):904–920, 1985.

- [31] E. Gamma, R. Helm, R. E. Johnson, and J. M. Vlissides. Design patterns: Abstraction and reuse of object-oriented design. In O. Nierstrasz, editor, ECOOP'93 - Object-Oriented Programming, 7th European Conference, Kaiserslautern, Germany, July 26-30, 1993, Proceedings, volume 707 of Lecture Notes in Computer Science, pages 406–431. Springer, 1993.
- [32] I. Garrigós, J. Pardillo, J. Mazón, and J. Trujillo. A conceptual modeling approach for OLAP personalization. In Conceptual Modeling - ER 2009, 28th International Conference on Conceptual Modeling, Gramado, Brazil, November 9-12, 2009. Proceedings, pages 401–414, 2009.
- [33] A. Giacometti, P. Marcel, E. Negre, and A. Soulet. Query recommendations for OLAP discovery driven analysis. In DOLAP 2009, ACM 12th International Workshop on Data Warehousing and OLAP, Hong Kong, China, November 6, 2009, Proceedings, pages 81–88, 2009.
- [34] M. Golfarelli, D. Maio, and S. Rizzi. The dimensional fact model: A conceptual model for data warehouses. Int. J. Cooperative Inf. Syst., 7(2-3):215–247, 1998.
- [35] M. Golfarelli and S. Rizzi. UML-based modeling for what-if analysis. In Data Warehousing and Knowledge Discovery, 10th International Conference, DaWaK 2008, Turin, Italy, September 2-5, 2008, Proceedings, pages 1–12, 2008.
- [36] M. Golfarelli and S. Rizzi. Data Warehouse Design: Modern Principles and Methodologies. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2009.
- [37] K. Hahn, C. Sapia, and M. Blaschka. Automatically generating OLAP schemata from conceptual graphical models. In Third ACM International Workshop on Data Warehousing and OLAP (DOLAP 2000), Washington, DC, USA, November 10, 2000, pages 9–16, 2000.

- [38] J. Han, M. Kamber, and J. Pei. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.
- [39] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis), 14(6):1189–1196, 2008.
- [40] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. ACM Queue, 10(2):30:30–30:55, Feb. 2012.
- [41] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. Management Information Systems Quarterly, 28(1):75–106, 2004.
- [42] M. Hilal. A proposal for self-service OLAP endpoints for linked RDF datasets. In P. Ciancarini, F. Poggi, M. Horridge, J. Zhao, T. Groza, M. C. Suárez-Figueroa, M. d’Aquin, and V. Presutti, editors, Knowledge Engineering and Knowledge Management - EKAW 2016 Satellite Events, EKM and Drift-an-LOD, Bologna, Italy, November 19-23, 2016, Revised Selected Papers, volume 10180 of Lecture Notes in Computer Science, pages 245–250. Springer, 2017.
- [43] M. Hilal and C. G. Schuetz. Semantic web analysis graphs: Guided multidimensional analysis of linked open data. In Y. E. Chan, M. Boudreau, B. Aubert, G. Paré, and W. Chin, editors, 27th Americas Conference on Information Systems, AMCIS 2021, Virtual Conference, August 9-13, 2021. Association for Information Systems, 2021.
- [44] M. Hilal, C. G. Schuetz, and M. Schrefl. Superimposed multidimensional schemas for RDF data analysis. In 2017 IEEE 14th International Scientific Conference on Informatics, pages 104–110. IEEE, 2017.
- [45] M. Hilal, C. G. Schuetz, and M. Schrefl. Using superimposed multidimensional schemas and OLAP patterns for RDF data analysis. Open Comput. Sci., 8(1):18–37, 2018.

- [46] M. Hilal, C. G. Schütz, and M. Schrefl. An OLAP endpoint for RDF data analysis using analysis graphs. In N. Nikitina, D. Song, A. Fokoue, and P. Haase, editors, Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017, volume 1963 of CEUR Workshop Proceedings. CEUR-WS.org, 2017.
- [47] J. Horkoff, D. Barone, L. Jiang, E. Yu, D. Amyot, A. Borgida, and J. Mylopoulos. Strategic business modeling: representation and reasoning. Software & Systems Modeling, 13(3):1015–1041, Jul 2014.
- [48] H. Hultgren. Modeling the Agile Data Warehouse with Data Vault. Brighton Hamilton, 2012.
- [49] W. Inmon. Building The Data Warehouse (4th Ed.). Wiley Publishing, 2005.
- [50] W. H. Inmon and D. Linstedt. Data Architecture: A Primer for the Data Scientist – Big Data, Data Warehouse and Data Vault. Morgan Kaufmann, Amsterdam, 2015.
- [51] W. H. Inmon, D. Strauss, and G. Neushloss. DW 2.0: The Architecture for the Next Generation of Data Warehousing. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.
- [52] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh. Management of context-aware preferences in multidimensional databases. In Third IEEE International Conference on Digital Information Management (ICDIM), November 13-16, 2008, London, UK, Proceedings, pages 669–675, 2008.
- [53] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh. Applying recommendation technology in OLAP systems. In Enterprise Information Systems, 11th International Conference, ICEIS 2009, Milan, Italy, May 6-10, 2009. Proceedings, pages 220–233, 2009.

- [54] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh. Preference-based recommendations for OLAP analysis. In Proceedings of the 11th International Conference on Data Warehousing and Knowledge Discovery, DaWaK '09, pages 467–478, Berlin, Heidelberg, 2009. Springer-Verlag.
- [55] H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh. A framework for OLAP content personalization. In Advances in Databases and Information Systems - 14th East European Conference, ADBIS 2010, Novi Sad, Serbia, September 20-24, 2010. Proceedings, pages 262–277, 2010.
- [56] L. Jiang, D. Barone, D. Amyot, and J. Mylopoulos. Strategic models for business intelligence. In M. Jeusfeld, L. Delcambre, and T.-W. Ling, editors, Conceptual Modeling – ER 2011: 30th International Conference, ER 2011, Brussels, Belgium, October 31 - November 3, 2011. Proceedings, pages 429–439, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [57] N. Juristo and A. M. Moreno. Introductory paper: Reflections on conceptual modelling. Data & Knowledge Engineering, 33(2):103 – 117, 2000.
- [58] R. S. Kaplan and D. P. Norton. The Balanced Scorecard - Measures That Drive Performance. Harvard Business Review, 70(1):71–79, January-February 1992.
- [59] R. S. Kaplan and D. P. Norton. Putting the Balanced Scorecard to Work. Harvard Business Review, 71(5):134–147, September-October 1993.
- [60] R. S. Kaplan and D. P. Norton. Using the Balanced Scorecard as a Strategic Management System. Harvard Business Review, 74(1):75–85, January-February 1996.
- [61] R. Kimball and J. Caserta. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data. John Wiley & Sons, 2004.

- [62] R. Kimball and M. Ross. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Wiley Publishing, 3rd edition, 2013.
- [63] R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, and B. Becker. The Data Warehouse Lifecycle Toolkit. Wiley Publishing, 2nd edition, 2008.
- [64] R. Kosara and J. D. Mackinlay. Storytelling: The next step for visualization. IEEE Computer, 46(5):44–50, 2013.
- [65] I. Kovacic, C. G. Schuetz, S. Schausberger, R. Sumeder, and M. Schrefl. Guided query composition with semantic OLAP patterns. In N. Augsten, editor, Proceedings of the Workshops of the EDBT/ICDT 2018 Joint Conference (EDBT/ICDT 2018), Vienna, Austria, March 26, 2018, volume 2083 of CEUR Workshop Proceedings, pages 67–74. CEUR-WS.org, 2018.
- [66] B. Lee, N. H. Riche, P. Isenberg, and S. Carpendale. More than telling a story: Transforming data into visually shared stories. IEEE Computer Graphics and Applications, 35(5):84–90, 2015.
- [67] D. Linstedt and K. Graziano. Super Charge Your Data Warehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. Createspace Independent Pub, 2011.
- [68] D. Linstedt and M. Olschimke. Building a Scalable Data Warehouse with Data Vault 2.0. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2015.
- [69] H. P. Luhn. A business intelligence system. IBM J. Res. Dev., 2(4):314–319, Oct. 1958.
- [70] S. Luján-Mora, J. Trujillo, and I.-Y. Song. A UML profile for multidimensional modeling in data warehouses. Data & Knowledge Engineering, 59(3):725 – 769, 2006.

- [71] P. Marcel. Log-driven user-centric OLAP. In 37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014, Opatija, Croatia, May 26-30, 2014, pages 1446–1451, 2014.
- [72] P. Marcel, R. Missaoui, and S. Rizzi. Towards intensional answers to OLAP queries for analytical sessions. In DOLAP 2012, ACM 15th International Workshop on Data Warehousing and OLAP, Maui, HI, USA, November 2, 2012, Proceedings, pages 49–56, 2012.
- [73] P. Marcel and E. Negre. A survey of query recommendation techniques for data warehouse exploration. In Actes des 7èmes journées francophones sur les Entrepôts de Données et l’Analyse en ligne, Clermont- Ferrand, France, EDA 2011, Juin 2011, pages 119–134, 2011.
- [74] S. T. March and A. R. Hevner. Integrated decision support systems: A data warehousing perspective. Decis. Support Syst., 43(3):1031–1043, Apr. 2007.
- [75] A. Maté and J. Trujillo. Tracing conceptual models’ evolution in data warehouses by using the model driven architecture. Computer Standards & Interfaces, 36(5):831–843, 2014.
- [76] A. Maté, J. Trujillo, and J. Mylopoulos. Specification and derivation of key performance indicators for business analytics: A semantic approach. Data & Knowledge Engineering, 108:30–49, 2017.
- [77] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In Machine Intelligence, pages 463–502. Edinburgh University Press, 1969.
- [78] MetaCase. MetaEdit+ Workbench 4.5 SR1 User’s Guide, 2009.
- [79] G. W. Mineau, R. Missaoui, and R. Godinx. Conceptual modeling for data and knowledge management. Data & Knowledge Engineering, 33(2):137 – 168, 2000.

- [80] R. Morgan, G. Grossmann, M. Schrefl, and M. Stumptner. Using VizDSL for modelling visualization processes. In 22nd IEEE International Enterprise Distributed Object Computing Workshop, EDOC Workshops 2018, Stockholm, Sweden, October 16-19, 2018, pages 212–215. IEEE Computer Society, 2018.
- [81] R. Morgan, G. Grossmann, M. Schrefl, and M. Stumptner. A model-driven approach for visualisation processes. In Proceedings of the Australasian Computer Science Week Multiconference, ACSW 2019, Sydney, NSW, Australia, January 29-31, 2019, pages 55:1–55:10. ACM, 2019.
- [82] R. Morgan, G. Grossmann, M. Schrefl, M. Stumptner, and T. Payne. VizDSL: A visual DSL for interactive information visualization. In J. Krogstie and H. A. Reijers, editors, Advanced Information Systems Engineering - 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings, volume 10816 of Lecture Notes in Computer Science, pages 440–455. Springer, 2018.
- [83] R. Morgan, G. Grossmann, and M. Stumptner. VizDSL: Towards a graphical visualisation language for enterprise systems interoperability. In 2017 International Symposium on Big Data Visual Analytics, BDVA 2017, Adelaide, Australia, November 7-10, 2017, pages 31–38. IEEE, 2017.
- [84] R. Morgan, G. Grossmann, M. Stumptner, and M. Schrefl. Modelling the semantics for model-driven interactive visualizations. In 23rd IEEE International Enterprise Distributed Object Computing Conference, EDOC 2019, Paris, France, October 28-31, 2019, pages 132–141. IEEE, 2019.
- [85] D. Murray. Tableau Your Data!: Fast and Easy Visual Analysis with Tableau Software. Wiley Publishing, 1st edition, 2013.

- [86] J. Mylopoulos. A perspective for research on conceptual modelling. In Proceedings of the 1980 Workshop on Data Abstraction, Databases and Conceptual Modeling, pages 167–170. ACM, 1980.
- [87] T. Neuböck. Analysis Process Modeling Notation For Business Intelligence (APMN4BI). Dissertation. Appendix A. Case Study: LEICON. Technical report, Johannes Kepler Universität Linz, Institut für Wirtschaftsinformatik – Data & Knowledge Engineering, 2022. <http://files.dke.uni-linz.ac.at/publications/pt2201/appendix.pdf>.
- [88] T. Neuböck. Analysis Process Modeling Notation For Business Intelligence (APMN4BI). Dissertation. Appendix B. Case Study: KOTI Kobra. Technical report, Johannes Kepler Universität Linz, Institut für Wirtschaftsinformatik – Data & Knowledge Engineering, 2022. <http://files.dke.uni-linz.ac.at/publications/pt2201/appendix.pdf>.
- [89] T. Neuböck. Analysis Process Modeling Notation For Business Intelligence (APMN4BI). Dissertation. Appendix C. Case Study: PVA. Technical report, Johannes Kepler Universität Linz, Institut für Wirtschaftsinformatik – Data & Knowledge Engineering, 2022. <http://files.dke.uni-linz.ac.at/publications/pt2201/appendix.pdf>.
- [90] T. Neuböck, B. Neumayr, T. Rossgatterer, S. Anderlik, and M. Schrefl. Multi-dimensional navigation modeling using BI analysis graphs. In ER Workshops, volume 7518 of Lecture Notes in Computer Science, pages 162–171. Springer, 2012.
- [91] T. Neuböck, B. Neumayr, M. Schrefl, and C. G. Schütz. Ontology-driven business intelligence for comparative data analysis. In eBISS, volume 172 of Lecture Notes in Business Information Processing, pages 77–120. Springer, 2013.

- [92] T. Neuböck and M. Schrefl. Modelling knowledge about data analysis processes in manufacturing. In Proceedings of the 15th IFAC/IEEE/IFIP/IFORS Symposium Information Control Problems in Manufacturing (INCOM 2015), May 11-13, 2015, Ottawa, Canada, volume 48 of IFAC-PapersOnLine, pages 277–282, 5 2015.
- [93] B. Neumayr, S. Anderlik, and M. Schrefl. Towards ontology-based OLAP: datalog-based reasoning over multidimensional ontologies. In DOLAP 2012, ACM 15th International Workshop on Data Warehousing and OLAP, Maui, HI, USA, November 2, 2012, Proceedings, pages 41–48, 2012.
- [94] B. Neumayr, M. Schrefl, and K. Linner. Semantic cockpit: An ontology-driven, interactive business intelligence tool for comparative data analysis. In ER Workshops 2011, Springer LNCS vol. 6999, pages 55–64, 2011.
- [95] B. Neumayr, C. Schütz, and M. Schrefl. Semantic enrichment of OLAP cubes: Multi-dimensional ontologies and their representation in SQL and OWL. In OTM Conferences, pages 624–641, 2013.
- [96] Object Management Group (OMG). Common warehouse metamodel (CWM) specification, 2003.
- [97] A. Olivé. Conceptual Modeling of Information Systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [98] OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011.
- [99] OMG. OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.4.1, August 2011.
- [100] OMG. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, August 2011.
- [101] OMG. Interaction Flow Modeling Language, Version 1.0, February 2015.

- [102] Oracle. Oracle business intelligence foundation suite. technical overview. <http://www.oracle.com/us/obiee-11g-technical-overview-078853.pdf>, 2011.
- [103] N. Raden. Guided and open-ended analytics: Serving the real users of business intelligence. White paper, Tableau Software, <https://www.tableau.com/sites/default/files/whitepapers/tableau-neil-raden-guided-open-ended-analytics.pdf>, 2009.
- [104] F. Ravat, O. Teste, R. Tournier, and G. Zurfluh. Algebraic and graphic languages for OLAP manipulations. *IJDWM*, 4(1):17–46, 2008.
- [105] R. Reiter. Artificial intelligence and mathematical theory of computation. chapter The Frame Problem in Situation the Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression, pages 359–380. Academic Press Professional, Inc., San Diego, CA, USA, 1991.
- [106] O. Romero and A. Abelló. A survey of multidimensional modeling methodologies. *IJDWM*, 5(2):1–23, 2009.
- [107] O. Romero, P. Marcel, A. Abelló, V. Peralta, and L. Bellatreche. Describing analytical sessions using a multidimensional algebra. In Data Warehousing and Knowledge Discovery - 13th International Conference, DaWaK 2011, Toulouse, France, August 29-September 2, 2011. Proceedings, pages 224–239, 2011.
- [108] C. Sapia. On modeling and predicting query behavior in OLAP systems. In S. Gatzju, M. A. Jeusfeld, M. Staudt, and Y. Vassiliou, editors, Proceedings of the Intl. Workshop on Design and Management of Data Warehouses, DMDW'99, Heidelberg, Germany, June 14-15, 1999, volume 19 of CEUR Workshop Proceedings, pages 2.1 – 2.10. CEUR-WS.org, 1999.
- [109] C. Sapia. PROMISE: predicting query behavior to enable predictive caching strategies for OLAP systems. In Y. Kambayashi, M. K. Mohania, and A. M. Tjoa, editors, Data Warehousing and Knowledge

- Discovery, Second International Conference, DaWaK 2000, London, UK, September 4-6, 2000, Proceedings, volume 1874 of Lecture Notes in Computer Science, pages 224–233. Springer, 2000.
- [110] C. Sapia, M. Blaschka, and G. Höfling. GraMMi: Using a standard repository management system to build a generic graphical modeling tool. In 33rd Annual Hawaii International Conference on System Sciences (HICSS-33), 4-7 January, 2000, Maui, Hawaii, USA, pages 1–10, 2000.
- [111] M. Schrefl, B. Neumayr, and M. Stumptner. The decision-scope approach to specialization of business rules: Application in business process modeling and data warehousing. In Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling (APCCM 2013), 2013.
- [112] C. Schuetz, L. Bozzato, B. Neumayr, M. Schrefl, and L. Serafini. Knowledge graph OLAP a multidimensional model and query operations for contextualized knowledge graphs. In Semantic Web, IOS Press, pages 1–35, 2020.
- [113] C. G. Schuetz, B. Neumayr, M. Schrefl, and T. Neuböck. Reference modeling for data analysis: The BIRD approach. Int. J. Cooperative Inf. Syst., 25(2):1–46, 2016.
- [114] C. G. Schuetz, S. Schausberger, I. Kovacic, and M. Schrefl. Semantic OLAP patterns: Elements of reusable business analytics. In H. Panetto, C. Debruyne, W. Gaaloul, M. P. Papazoglou, A. Paschke, C. A. Ardagna, and R. Meersman, editors, On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part II, volume 10574 of Lecture Notes in Computer Science, pages 318–336. Springer, 2017.
- [115] C. G. Schuetz and M. Schrefl. Towards formal strategy analysis with goal models and semantic web technologies. In S. de Cesare

- and U. Frank, editors, Advances in Conceptual Modeling - ER 2017 Workshops AHA, MoBiD, MREBA, OntoCom, and QMMQ, Valencia, Spain, November 6-9, 2017, Proceedings, volume 10651 of Lecture Notes in Computer Science, pages 144–153. Springer, 2017.
- [116] C. G. Schütz, B. Neumayr, and M. Schrefl. Business model ontologies in OLAP cubes. In Advanced Information Systems Engineering - 25th International Conference, CAiSE 2013, Valencia, Spain, June 17-21, 2013. Proceedings, pages 514–529, 2013.
- [117] C. G. Schütz and M. Schrefl. Customization of domain-specific reference models for data warehouses. In 18th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2014, Ulm, Germany, September 1-5, 2014, pages 61–70, 2014.
- [118] B. Silver. BPMN Method and Style: With BPMN Implementer's Guide. Cody-Cassidy Press, 2011.
- [119] V. Stefanov, B. List, and B. Korherr. Extending UML 2 activity diagrams with business intelligence objects. In Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2005), pages 53–63. Springer Verlag, 2005.
- [120] D. Steiner, B. Neumayr, and M. Schrefl. Judgement and analysis rules for ontology-driven comparative data analysis in data warehouses. In 11th Asia-Pacific Conference on Conceptual Modelling, APCCM 2015, Sydney, Australia, January 2015, pages 71–80, 2015.
- [121] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. IEEE Transactions on Visualization and Computer Graphics, 8(1):52–65, Jan. 2002.
- [122] C. Stolte, D. Tang, and P. Hanrahan. Multiscale visualization using data cubes. IEEE Trans. Vis. Comput. Graph., 9(2):176–187, 2003.

- [123] V. C. Storey, J. Trujillo, and S. W. Liddle. Research on conceptual modeling: Themes, topics, and introduction to the special issue. Data & Knowledge Engineering, 98:1–7, 2015.
- [124] Tableau Software. (c) 2003-2017 Tableau Software. 1621 N 34th St., Seattle, WA 98103, US, <http://www.tableau.com>, 2017.
- [125] T. Thalhammer and M. Schrefl. Realizing active data warehouses with off-the-shelf database technology. Softw., Pract. Exper., 32(12):1193–1222, 2002.
- [126] T. Thalhammer, M. Schrefl, and M. Mohania. Active data warehouses: complementing OLAP with analysis rules. Data & Knowledge Engineering, 39(3):241 – 269, 2001.
- [127] TIBCO Spotfire. TIBCO Spotfire guided analytics. BeyeNETWORK, http://www.b-eye-network.com/offers/spotfire/tibco_spotfire_guided_analytics.pdf, 2007.
- [128] J. Trujillo, J. Gómez, and M. Palomar. Modeling the behavior of OLAP applications using an UML compliant approach. In Advances in Information Systems, pages 14–23. Springer, 2000.
- [129] J. Trujillo, M. Palomar, and J. Gómez. Applying object-oriented conceptual modeling techniques to the design of multidimensional databases and OLAP applications. In Web-Age Information Management, First International Conference, WAIM 2000, Shanghai, China, June 21-23, 2000, Proceedings, pages 83–94, 2000.
- [130] J. Trujillo, M. Palomar, J. Gomez, and I.-Y. Song. Designing data warehouses with OO conceptual models. IEEE Computer, 34(12):66–75, 2001.
- [131] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of

- issues and approaches. Data & Knowledge Engineering, 47(2):237–267, Nov. 2003.
- [132] W. M. P. van der Aalst and A. J. M. M. Weijters. Process mining: A research agenda. Computer in Industry, 53(3):231–244, Apr. 2004.
- [133] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. Conceptual modeling for ETL processes. In Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP, DOLAP '02, pages 14–21. ACM, 2002.
- [134] H. J. Watson and B. H. Wixom. The current state of business intelligence. IEEE Computer, 40(9):96–99, 2007.
- [135] Y. Zhang, C. Ordonez, J. García-García, and L. Bellatreche. Optimization of percentage cube queries. In Proceedings of the Workshops of the EDBT/ICDT 2017 Joint Conference (EDBT/ICDT 2017), Venice, Italy, March 21-24, 2017., 2017.

Curriculum Vitae



Dipl.-Ing. Thomas Neuböck is a senior managing consultant of solvistas GmbH and a shareholder and CEO of solvistas Group GmbH, Graben 18, 4020 Linz, AUSTRIA. He was born in Upper Austria in 1969, is married and father of two children. From 1989 to 1997, he studied Informatics and Business Informatics at the Johannes Kepler University Linz and received the degree of a Diplomingenieur. He is a member of IEEE since 1992 and a certified Project Management Professional (PMP) of the Project Management Institute (PMI) since 2012. Thomas Neuböck developed business software and database applications for manufacturing companies, energy suppliers, trading, and service companies. Furthermore, he was trainer for software development. Since 2004 his job engagement is focused on business intelligence (BI), data warehousing (DWH), and data science. He works as an architect, consultant, business analyst, and project manager in the area of BI and DWH. In this functions, he conducts and accompanies long-term projects of, among others, Austrian and German public insurance organizations and manufacturing companies.