Implementing Judgement and Analysis Rules for Comparative Data Analysis in Oracle

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Masterstudium

Wirtschaftsinformatik

Eingereicht von:
Dieter Steiner

Angefertigt am:
Institut für Wirtschaftsinformatik - Data & Knowledge Engineering

Beurteilung:
o.Univ.-Prof. DI Dr. Michael Schrefl

Mitwirkung:
Dr. Bernd Neumayr

Linz, Mai 2014

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, Mai 2014

Dieter Steiner

Abstract

In this work, we present the implementation of a prototype that extends a conventional data warehouse with judgement and analysis rules by using a commercial database system. This thesis is based on ideas and concepts [Neuböck et al., 2013; Schrefl et al., 2013] developed in the Semantic Cockpit project. The main contribution of this thesis is the implementation of these ideas and concepts. Analysis rules aid analysts in conducting routine and semi-routine decision tasks. Judgement rules are used to externalise knowledge about insights gained during data analysis. Both rule types aid to facilitate data analysis tasks. The presented prototype is part of the Semantic Cockpit research project in which an ontology-driven business intelligence approach for comparative data analysis has been developed, and is based on the developed data warehouse and multi-dimensional ontology.

Kurzfassung

Diese Arbeit beschreibt die Implementierung eines Prototyps, der es auf Basis eines kommerziellen Datenbankmanagementsystems ermöglicht ein konventionelles Data Warehouse mit Judgement-Regeln und Analyse-Regeln zu erweitern. Diese Arbeit basiert auf Ideen und Konzepten [Neuböck et al., 2013; Schrefl et al., 2013], die im Rahmen des Semantic Cockpit Forschungsprojektes entwickelt wurden. Der wesentliche Beitrag dieser Arbeit besteht in der Implementierung dieser Ideen und Konzepte. Analyse-Regeln helfen Analytikern und Analytikerinnen wiederkehrende Entscheidungen zu treffen. Judgement-Regeln werden verwendet um Fachwissen zu externalisieren und im Rahmen der Datenanalyse generiertes Wissen zu explizieren. Beide Regeltypen zielen darauf ab die Aufgabe der Datenanalyse zu erleichtern. Der in dieser Arbeit vorgestellte Prototyp ist Teil des Semantic Cockpit Projektes in dem ein ontologie-basierter Business Intelligence-Ansatz zur vergleichenden Datenanalyse entwickelt wurde. Der Prototyp basiert auf dem Data Warehouse und der multi-dimensionalen Ontologie, die im Rahmen dieses Projektes entwickelt wurden.

Contents

1 Introduction

Data warehouses are dedicated to integrating data from online transaction processing (OLTP) systems in order to provide a comprehensive data base for managerial decisions. Analysts perform interactive analysis on the data warehouse with online analytical processing (OLAP) tools in order to find solutions for decision tasks. According to Inmon [1992], a data warehouse is a subject-oriented, integrated, nonvolatile, and time variant collection of data designed to support management decisions.

A data warehouse organises data using the dimensional modeling approach [Kimball and Strehlo, 1994], which classifies data into measures and dimensions. Measures are the basic units of interest for data analysis. Dimensions allow evaluating measures from different perspectives, for example, time, customer, or product. Each dimension is organised hierarchically with different dimension levels, which offers the possibility to view measures at different granularities. For example, granularities for dimension time can be day, month, and year. The aggregation of measures up to a certain dimension level creates a multi-dimensional view, which is also known as data cube or cube for short [Datta and Thomas, 1999] [Agrawal et al., 1997] [Gray et al., 1997].

Business analysts explore data cubes with the purpose of finding meaningful performance indicators that can be used to support business decisions. Measure values contained in a cube are usually numeric values. These values can be evaluated with respect to other values, for example, reference values of performance targets, in order to increase their meaningfulness. Similarly, analysts can perform comparative data analysis by comparing measure values of different groups of data. For example, a comparative analysis might compare aggregated measure values of the current year with values of the previous year. In comparative analysis situations, scores compare the measure values of different cubes, thus capture the results of a comparison. This work focuses on these is-to-is comparisons of data.

In the course of the analysis process the analyst obtains additional insights on the underlying data. These insights can be used either as starting points for further investigations, for example, using data mining techniques, or as rationale for business decisions. Further, the generated knowledge can be employed to guide future analysis tasks.

Comparative data analysis is a complex and time-consuming task. As the goal of data analysis is to gain new insights, it is an *"explorative, iterative and incremental process"* [Neuböck et al., 2013, p. 36]. The main challenges of a business analyst are to identify relevant comparison groups and to define the relevant scores for showing their relationships. Specific domain knowledge can help the analyst during this process.

Traditional data warehouses do not provide sufficient support for comparative data analysis. In order to conduct comparative analysis, specific domain knowledge is needed for the formulation of analysis queries. A repository that defines the semantics of business terms can aid analysts with the formulation of comparative data analysis queries as the use of defined business terms allows the analyst to abstract from the concrete definitions. It allows business analysts to reuse previously defined concepts in order to formulate similar yet different queries. Further, the use of business terms improves readability of query expressions for other users. Current data warehouses do not provide such a repository. Therefore, the business analyst has to rely on his expert knowledge, or an external knowledge base, in order to formulate correct business terms during query creation.

In the Semantic Cockpit (semCockpit) research project a multi-dimensional ontology (MDO) provides a repository of business term definitions. These business terms are specifically designed for use in an OLAP system. Business terms in the MDO are unambiguous and are hierarchically ordered in subsumption hierarchies. Further, business terms are first-class citizens in semCockpit. This allows analysts to utilise the defined business terms for data analysis. A comprehensive description of the semCockpit approach can be found in Neuböck et al. [2013].

The main objective of semCockpit is to support the comparative data analysis process. Scores, comparative concepts, and comparative cubes are used to make comparisons first-class citizens in semCockpit. Scores capture the results of a comparison of different measure values. Comparative concepts specify relevant comparison groups and their relationship. A comparative cube is a data cube that captures the result of a comparative analysis and is therefore an explicit representation of a comparative data analysis situation.

In Thalhammer et al. [2001] and Thalhammer and Schrefl [2002], analysis rules are specified in the context of data warehousing as means to (semi-) automate routine and semi-routine decision tasks. Analysis rules have been inspired by active database systems, which employ the event-condition-action (ECA) paradigm in order to automate recurring tasks in online transaction processing (OLTP) databases [Thalhammer et al., 2001, p. 242]. A different, yet in some aspects similar form of rules for data warehouses are judgement rules, which are introduced in Neumayr et al. [2011] and further elaborated in Neuböck et al. [2013].

In this work the implementation of a rule engine prototype is described, which implements the core functionalities of judgement and analysis rules, as described in

Neuböck et al. [2013], on top of the semCockpit system prototype. Judgement rules represent stored knowledge for facts of comparative analysis situations whose score values are extraordinarily low or high. They provide some judgement or possible explanations for these striking values. Analysis rules are likewise defined over facts of a comparative analysis situation. Analysis rules define actions that should be executed depending on the score values of a fact. For example, a possible action is that the facts in question are reported. Analysis rules allow hierarchical analysis of the underlying data along a defined analysis path, and therefore require specific evaluation strategies with respect to inheritance and overriding.

The remainder of this work is organised as follows: In chapter 2 the Semantic Cockpit project and its underlying concepts and ideas are presented. This includes the semCockpit data warehouse (semDWH) and the MDO together with the MDO database (MDO-DB). Further, MDO concepts, comparative concepts, measures, scores, and comparative cubes, which are used for comparative data analysis, are described. This chapter also discusses the notion of generic measures and generic scores as defined by the semCockpit approach. This includes the introduction of generic comparative cubes as means for specifying a generic comparative analysis situation.

Chapter 3 introduces the conceptual basis of the rule engine prototype. The two different rule types, judgement and analysis rules, are discussed. Two different evaluation strategies, prerogative and presumed, which define the hierarchical evaluation of analysis rules are discussed in detail. After introducing the base rule types we present a generic extension of rules, which allows to define rules not only for concrete but also for generic comparative analysis situations.

The implementation of the prototype system is presented in chapter 4. First, we give a brief overview on the semCockpit system architecture underlying the rule engine prototype. Following this, we present our implementation of rules within the MDO-DB as well as the corresponding representation in the semDWH. Next, the process for mapping MDO rule definitions into semDWH representations is described. After that, we provide our implementation for rule evaluation. In addition, we provide the results of a preliminary performance study. The presented results highlight the parts of the prototype with the highest performance impact. Finally, we discuss the state of the current rule engine implementation and point at relevant features that were not within the scope of this work.

Finally, chapter 5 gives a summary of the presented work and provides an outlook for further enhancements and applications of the presented rule engine prototype. Specifically, we take a look at guidance rules as introduced in Neuböck et al. [2013], which might be implemented similar to the rule types covered by the current prototype.

2 The Semantic Cockpit Approach

This chapter provides an overview of the concepts of the Semantic Cockpit (semCockpit) research project for which the rule engine prototype described in this work has been developed. The rule prototype is part of the semCockpit proof-of-concept prototype. A comprehensive description of the ontology-driven approach employed can be found in Neuböck et al. [2013]. After giving a general overview on the semCockpit approach and related work in the context of semantic data warehousing, the specifics of the semCockpit data warehouse (semDWH) are explained. Then, we present the multi-dimensional ontology (MDO) and the MDO database (MDO-DB). Further, we provide the definitions of the different types of concepts that can be defined in the MDO. After introducing the basic concept types we provide a more detailed overview of measures, scores, cubes, and comparative cubes, which are used for defining comparative analysis situations. We also discuss the notion of generic measures and scores as well as generic comparative cubes for specifying generic comparative analysis situations. Finally, we describe the mapping component used for communication between the semDWH and the MDO-DB.

One of the main objectives of semCockpit is to fully support business analysts in conducting comparative data analysis. Traditional business intelligence (BI) tools leave the definition of domain-specific business terms to the analysts. This results in business terms being implicitly defined within query applications. In contrast, semCockpit provides a central repository in form of the MDO, which contains unambiguous definitions together with the semantics of business terms. This repository is organised as a multi-dimensional ontology, which is stored in the MDO-DB, and contains explicit representations of domain-specific concepts. Analysts can employ business terms defined in the MDO for the formulation of OLAP queries, which facilitates the query-creation process. The semCockpit approach especially supports the definition of concepts dedicated for use in comparative analysis queries.

Ontology-based business intelligence approaches have received considerable attention and several complementary applications have been proposed. Ontologies can be utilised during the data warehouse design process for automating tasks of the data warehouse schema generation [Romero and Abelló, 2007; Khouri and Ladjel, 2010; Sciarrone et al., 2009; Nebot et al., 2009]. Nebot et al. [2009] provide a framework for designing multi-dimensional analysis models, which they call Multidimensional Integrated Ontology (MIO), over the semantic annotations stored in a semantic data warehouse. Their approach allows the analysis of data by using traditional OLAP operators [Nebot and Llavori, 2012; Nebot et al., 2009]. For further discussion of related work also see Neuböck et al. [2013] and Romero and Abelló [2013].

Using ontologies for querying databases is also an active area of research. See the Related Work sections of Neumayr et al. [2011] and Neumayr et al. [2013]. Some work focuses on using ontologies as global, integrated view on top of different heterogenous data sources. One example of this approach is the MASTRO system [Calvanese et al., 2011], which uses an ontology as query interface in order to provide an integrated conceptual view over underlying data sources. Spahn et al. [2008] define an approach enabling ontology-based querying in business intelligence. In order to facilitate query formulation for end-users they present the Semantic Query Designer (SQD), which provides a graphical user interface for navigation within the business ontology and for the definition of queries. Spahn et al. specifically address the applicability of their approach with respect to multi-dimensional data analysis as future research direction. Lim et al. [2007] employ a related approach by integrating domain knowledge as concept definitions in terms of virtual views. These views enrich relational data with information derived from a domain ontology and provide a query interface for semantic queries. The semCockpit approach similar to these approaches uses an ontology as conceptual view over the data repository. In contrast to previous work, the semCockpit approach focuses on the specifics of OLAP and especially comparative data analysis.

The core semCockpit business logic is split into three main components. First, the semDWH holds all business data like any conventional data warehouse. The data in the semDWH is stored in a specific form in order to allow interoperability with other semCockpit components. Second, the MDO-DB provides the repository of business terms. Finally, the MDO-DWH Mapper is responsible for managing the communication between the semDWH and the MDO-DB. The MDO-DWH Mapper generates the correct SQL statements, typically a create view statement, in order to reproduce each MDO business term as object in the semDWH. Additional components are built on top of this core and enhance the functionality of the semCockpit system in different areas.

Figure 2.1 shows the different components of the semCockpit system. The core components, semDWH, MDO-DWH Mapper, and MDO, form the foundation for additional components. For presentation of the prototype a basic frontend showcasing the characteristic features of the approach has been developed. This thesis focuses on the conceptualisation and implementation of the rule engine component and the therefore necessary extensions to core components.

The rule engine component heavily depends on the underlying implementation of

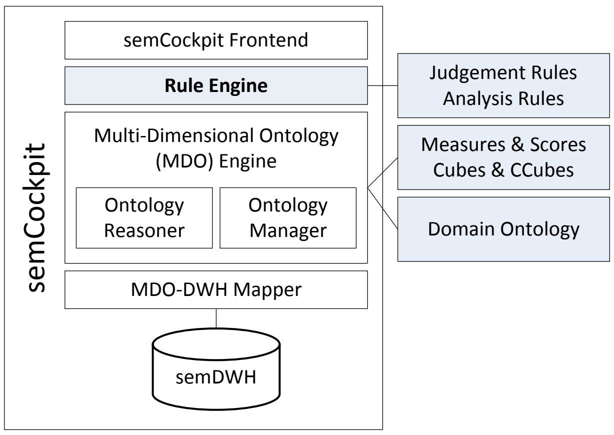

Figure 2.1: The semCockpit stack

the semCockpit system prototype. To make this thesis self-contained, we discuss the relevant concepts of the core components in the following sections in order to provide a grounding for the conceptualisation of the rule engine. Neumayr et al. [2013] provide the language specification for the realisation of MDO constructs. Additionally, a comprehensive description of the semCockpit approach can be found in Neuböck et al. [2013].

2.1 semCockpit Data Warehouse

In this section we introduce the structure and specifics of the semCockpit data warehouse (semDWH).

A data warehouse (DWH), or sometimes also referred to as multi-dimensional database, usually organises data based on multiple dimensions using the dimensional modeling approach introduced by Kimball and Strehlo [1994]. The basic conceptual construct of a data warehouse is a data cube containing dimensions, attributes, and measures [Datta and Thomas, 1999, p. 291]. A data cube provides a multi-dimensional view over the data in the DWH. A DWH consists of multi-dimensional facts. Facts are usually represented in fact tables, which contain the actual measured data (measures) as well as foreign keys, which refer to each of the fact's dimensions [Datta and Thomas, 1999, p. 292]. A specific fact within a data warehouse can be identified by a set of dimension nodes. The nodes of each dimension form a hierarchy consisting of different dimension levels. Each dimension node, in addition to its place within the dimension hierarchy, can define non-dimensional attributes, which further describe the node. All nodes of the same dimension level define the same set of non-dimensional attributes. The facts of a data warehouse that are represented in fact tables refer for each dimension to a node of the most fine-grained dimension level.

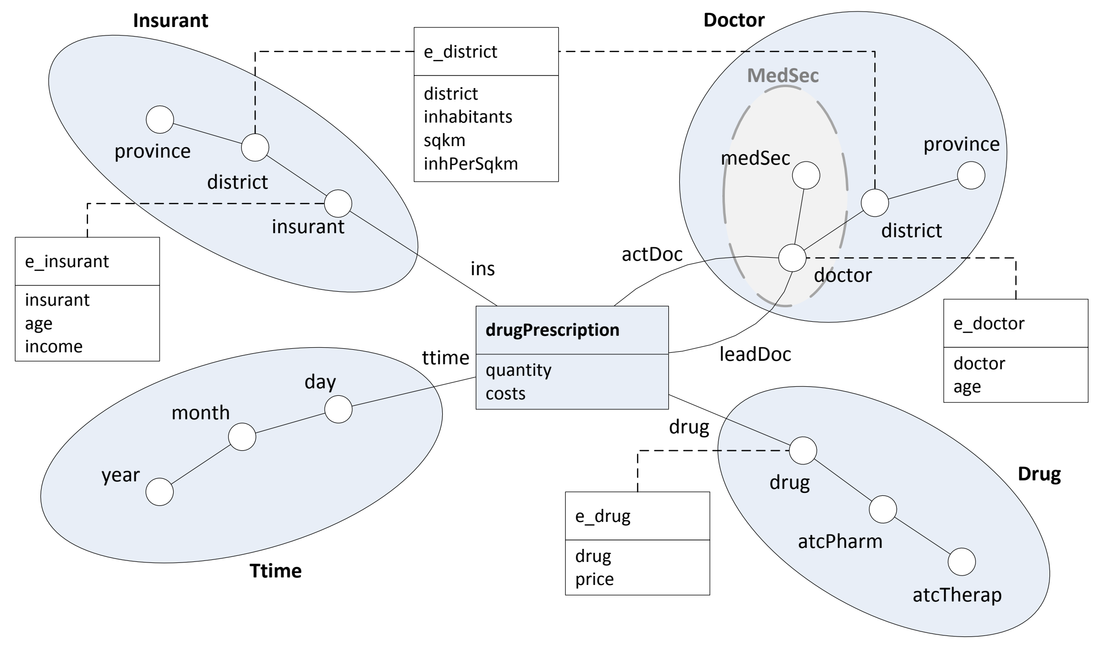

Figure 2.2: semCockpit DWH schema

Example 1 (semDWH). Figure 2.2 shows a simplified excerpt of the schema of a semCockpit data warehouse, adapted from Neuböck et al. [2013, p. 5], represented by using a slight variation of the dimensional fact model [Golfarelli et al., 1998]. The model shows the semDWH structure for drug prescription transactions. The depicted dimensions are Insurant, Doctor, Drug, and Ttime and constitute the different dimensions of a drug prescription. The different dimension levels for each dimension are shown. Top levels of dimensions containing for each dimension a specific all-node are omitted. For example, the level hierarchy of dimension Insurant consists of the dimension levels insurant, for single insurants; district, for aggregating all insurants that reside in the same district; province, for aggregating insurants residing in the same province; and all, for aggregating all insurants. The dimension level medSec of dimension Doctor describes the medical section of a doctor. An example for a node of this dimension level might be general practitioner. Dimension Drug organises drugs by a subset of the Anatomical Therapeutic Chemical Classification (ATC-Classification) system. Level atcPharm denotes the therapeutic or pharmacological subgroup of drugs and level atcTherap the therapeutic main group. Note, that additional levels are defined by the ATC-Classification that are omitted in the depicted representation. The fact class drugPrescription, representing the collected data of drug prescription transactions, contains the base measures quantity and costs. The used dimension roles for drugPrescription facts are ins (insurant), leadDoc (leading doctor), actDoc (acting doctor), drug and ttime.

In the semDWH data model, entity classes are used to specify the non-dimensional attributes of dimension nodes. A dimension level can refer to an entity class so that each node of that level refers to a different entity of the referred entity class. An entity class may be referred to by several dimension levels of different dimensions, thus avoiding redundancy.

Example 2 (Entity Class). The entity class e_district contains the attribute district, which is used as external identifier, and a set of non-dimensional attributes, which describe the entity. The attribute inhabitants denotes the total population of a district entity and attribute sqkm describes the area of the entity. Attribute inhPerSqkm denotes the population density of a district and can be derived from the two other non-dimensional attributes. In the depicted DWH schema, level district of each of the two dimensions Insurant and Doctor refers to instances of the entity class e_district so that nodes of the different dimensions at the respective district level may refer to the same entity of e_district.

Dimension spaces are used to specify a domain for measures and multi-dimensional concepts and are another inherent concept of the semDWH data model. A dimension space consists of a set of dimension roles and for each dimension role a specific level or level range. A dimension role is defined by a dimension together with a role name. Dimension spaces can contain the same dimension multiple times as long as they play different dimension roles. Points within a dimension space are identified by the nodes of the defined dimension roles. In order to fully identify a point within a dimension space, for each dimension role a corresponding node has to be specified. A specific point is in the interpretation of a dimension space, if it defines a corresponding node for each of the dimension roles defined by the dimension space.

A dimension space can, for each dimension role, be restricted to either a specific granularity specified by a dimension level, or a granularity range defined by a closed interval of two dimension levels. Dimension spaces that are restricted to a specific dimension level for each dimension role are called *monogranular dimension spaces*. Conversely, if a dimension space is either restricted to a granularity range or unrestricted for one or more of its dimension roles, it is referred to as *multigranular dimension space*.

Example 3 (Dimension Space). The excerpt of the semDWH schema in figure 2.2 depicts the base dimension space drugPrescription defined by the five dimension roles insurant, ttime, drug, leadDoc and actDoc. The roles leadDoc and actDoc are defined on the same underlying dimension Doctor. A base dimension space in the semDWH is not restricted to a specific granularity or granularity range. Base dimension spaces can be restricted to a certain granularity or granularity range by specifying a level or level range for each of the base dimension's dimension roles. For example, drugPrescription [ins:district..province, leadDoc:all, actDoc:all, drug:all, ttime:year] denotes a multigranular restriction of the drugPrescription dimension space. The restriction for dimension role ins denotes a level range restriction and is interpreted as a restriction to all nodes that belong either to level district or province, or to a level that is situated between these two levels in the dimension hierarchy. All other dimension roles are restricted to a single dimension level. Note, that a restriction to the dimension level all is also possible. If the level range for a specific dimension role should not be restricted, the designation of the level range for this dimension role can be omitted. Therefore, drugPrescription [ins:district..province] denotes the restriction of the base dimension space drugPrescription to the levels district and province for dimension role ins with all other dimension roles being unrestricted. A specific point in a dimension space, for example, ⟨ttime:20130110, ins:mrErnst, leadDoc:drMayer, actDoc:drFalkner, drug:paracet500⟩, may be further described by measures.

Each measure in semCockpit is defined on a dimension space. The dimension space of a fact class and its containing base measures is monogranular and, for each dimension role of the dimension space, defined on the most fine-grained dimension level. Measures that are derived from one or more base measures, for example, SUM(costs), can be defined on freely chosen dimension spaces and are not restricted to the finest granularity. Note, however, that logical restrictions to the possible granularities of a dimension space might apply depending on the definition of the derived measure.

Example 4 (Dimension Space of Base Measure). The base dimension space drugPrescription is unrestricted for each of its dimension roles. The base measure quantity is defined on a restriction of the dimension space drugPrescription. Dimension space drugPrescription restricted to the most fine dimension level on each dimension role, that is, drugPrescription [ttime:day, ins:insurant, actDoc:doctor, leadDoc:doctor, drug:drug], is the monogranular-restricted dimension space of the base measure quantity. The same dimension space restriction applies to the second measure defined by the drugPrescription fact class, costs.

A dimension defines roll-up hierarchies for its dimension levels. In some cases a single dimension defines more than one hierarchy. These different hierarchies are usually treated as alternative hierarchies, so that each point is identified by exactly one node for each dimension role independent of the hierarchy the node is in. However, the semDWH data model also supports a different approach that allows for a joint integration of different hierarchies. In order to do so a dimension space defines for each additional hierarchy an additional hierarchy-specific dimension role. A point in the dimension space is then identified by one node for each conventional dimension role as well as one node for each hierarchy-specific dimension role.

Example 5 (Roll-up Hierarchies). In figure 2.2 dimension Doctor defines the hierarchy MedSec as named hierarchy. Therefore, hierarchy-specific dimension roles leadDocMedSec and actDocMedSec can be used to define dimension spaces with parallel roll-up paths. For example, a dimension space definition might include the dimension role actDoc as well as the hierarchy-specific dimension role actDocMedSec, leading to the following behaviour: When actDoc refers to a node at level doctor, then actDocMedSec refers to the same node. When actDoc refers to a node at level district or province, then actDocMedSec refers to some node at level medSec or to the hierarchy-specific all-node. Therefore, a query might select for example all doctors from province Upper Austria with a medical section of general practitioner. Finally, when actDoc refers to the role specific all-node, then actDocMedSec can either refer to its all-node as well, or to some node at level medSec.

A semCockpit data warehouse can be created in two different ways. First, the semDWH can be built from scratch like any other data warehouse. Second, it is possible to transform an existing data warehouse into a semDWH by defining an extract, transform, load (ETL) process that maps the existing data warehouse structure to the structure of a semDWH.

2.2 Multi-Dimensional Ontology

In this section the concept of a multi-dimensional ontology (MDO) is presented. This includes the definitions of the different concept types that can be defined in the MDO and stored in the MDO database (MDO-DB).

MDO describes an ontology containing an arbitrary number of defined business terms and the meaning thereof. A core feature of the MDO is the possibility to translate MDO concept definitions into SQL statements for querying the semDWH. Additionally, concepts can be translated into Web Ontology Language (OWL) statements for determining subsumption hierarchies and disjointness by OWL reasoners.

The main motivation behind the MDO is to enable the definition of business terms that can be employed during data analysis in the semCockpit data warehouse (semDWH). In the MDO, different types of concepts can be defined, whereby different types are defined over different types of objects within the semDWH. For the following semDWH objects MDO concepts can be defined: (1) entities of an entity class (*entity concept*), (2) nodes of a dimension (*dimensional concept*), (3) points of a dimension space (*multi-dimensional concept*), and (4) pairs of points of a comparative dimension space (*comparative concept*) [Neuböck et al., 2013, p. 13].

Each concept defines a *signature* and a *membership condition.* The *signature* of a concept indicates the type of individuals over which it is defined. Depending on the concept type the signature might be expressed by an entity class, a dimension, a dimension space, or a comparative dimension space. Therefore, the signature of an entity concept consists of the concept name and the entity class over which it is defined. The signature of a dimensional concept consists of its name and a dimension. Additionally, the dimension in the signature of a dimensional concept may be restricted to nodes of a specific dimension level or level range. The signature of a multi-dimensional concept is given by its name and a dimension space. This dimension space might be restricted to a specific granularity or granularity range. The signature of a comparative concept comprises next to its name, a comparative dimension space (comparison space), defined by a pair of dimension spaces. The dimension spaces defined by the comparative dimension space might be granularity or granularity-range restricted as well. Concept expressions provide the *membership condition* for concepts and specify which individuals are in the interpretation of a concept. The signature of a concept does not need to be explicitly specified as it can also be derived from the concept expression.

An entity concept is interpreted by a subset of the entities of an entity class. The concept expression specifies the selection condition and defines which entities are in the interpretation of the concept. Entity concepts are the simplest form of MDO concepts as entity classes are the most basic structures over which a concept can be defined. As entity classes can be referenced by multiple dimensions, an entity concept cannot be directly used in the context of a multi-dimensional analysis situation. Entity concepts are used as basic building blocks that are referenced by the concept expression definitions of other MDO concept types.

The following ways of providing the concept expression of an entity concept are described in Neuböck et al. [2013, p. 14]: (1) outside the MDO, that is, as primitive in the semDWH; (2) by a single entity of an entity class; (3) by an enumeration of entities; (4) by a boolean expression of attribute-value-comparisons so that all entities that satisfy the comparison are in the interpretation of the concept; and (5) by a SQL statement over an entity table of the underlying semDWH.

Example 6 (Entity Concept). The concept signature urbanDistrict (e_district) describes an entity concept defined on entity class e_district with the associated name urbanDistrict. The concept expression of this concept might be defined by a boolean expression on the entity class attribute inhPerSqkm, for example, inhPerSqkm > 500, so that all districts that satisfy the expression condition are in the interpretation of the concept. Note, that the concrete definition of what constitutes an urban district might vary for different analysts. The explicit definition in the MDO ensures that the concept urbanDistrict can be uniformly applied during data analysis without the analysts having to know its exact definition. Concept linz (e_district) describes an entity concept defined by the single district entity Linz-Stadt. The statutory cities of the province of Upper Austria might also be defined as entity concept. This can be achieved through an enumeration of the statutory cities Linz-Stadt, Wels-Stadt, and Steyr-Stadt in the concept expression of the concept statutoryUpperAustria (e_district).

Dimensional concepts are defined in the context of a dimension. Therefore, a dimensional concept is also valid in the context of dimension roles that reference the dimension over which the concept is defined. Like entity concepts, dimensional concepts are used as building blocks for the definition of other concept types that can then be used in analysis queries.

As described in Neuböck et al. [2013, p. 14f.], the concept expression of a dimensional concept is provided in one of the following ways: (1) as primitive outside the MDO, (2) by a node, (3) by a reference to an entity concept, (4) by hierarchy expansion of a dimensional concept, (5) by level range restriction of a dimensional concept, (6) by intersection of dimensional concepts or union of concepts defined for the same level range, (7) as complement of two dimensional concepts, (8) as ⟨node⟩concept[level-or-level range]-expression, or (9) by a SQL view over the semDWH.

Example 7 (Dimensional Concept). The concept signature insurantUrbanDistrict (Insurant[district]) describes a dimensional concept defined on dimension Insurant restricted to the dimension level district with the associated name insurantUrbanDistrict. The concept expression of this concept might be defined by a reference to the entity concept urbanDistrict. This concept comprises all districts contained in the referenced entity concept. Another dimensional concept insurantInUrbanDistrict (Insurant[insurant..district]) contains all nodes specified by the concept insurantUrbanDistrict (Insurant[district]) as well as all nodes of the dimension Insurant that roll up to a node that is in the interpretation of insurantUrbanDistrict. The concept expression of insurantInUrbanDistrict can be defined as hierarchy expansion of the concept insurantUrbanDistrict.

A multi-dimensional concept (MDC) is defined over a dimension space. Each dimension role of the dimension space can be restricted to a level or level range. The interpretation of a MDC is the set of points contained in the concept's dimension space that satisfy the concept definition for each of its dimension roles.

The concept expression of a multi-dimensional concept, according to Neuböck et al. [2013, p. 16], can be provided in one of the following ways: (1) as primitive outside the MDO, (2) by a specific point, (3) by reference to a dimensional concept for one of the MDC's dimension roles, (4) by hierarchy expansion of a MDC, (5) by granularity restriction of a MDC, (6) by intersection of MDCs or union of MDCs defined for the same dimension roles and the same granularities, (7) by complement of a MDC, (8) as ⟨point⟩concept[granularity-or-granularity range]-expression, (9) by a SQL view over the semDWH, or (10) by a boolean expression over measure-value-comparisons of measures applied to a point, so that the points that satisfy the expression are in the interpretation of the MDC. Further, each dimension space itself can also be used as MDC.

Example 8 (Multi-dimensional Concept). The MDC with signature insInUrbanDistrict (drugPrescription[ins:insurant..district]) should include all points that refer to some urban district in the dimension role ins as well as the points that roll up to such a point. The concept expression of this concept might be given by a reference to the dimensional concept insurantInUrbanDistrict for the dimension role ins. The MDC insInUrbanDistrict can be restricted by defining a new MDC. MDC insUrban-District might be defined by restricting insurantInUrbanDistrict to the dimension level district for dimension role ins. Therefore, the signature of the resulting concept is insUrbanDistrict (drugPrescription[ins:district]) whereby unrestricted dimension roles are omitted.

Comparative concepts (CC) are used to define point-pairs for comparative data analysis. A comparative concept consists of two groups of points, the group of interest (GoI) and the group of comparison (GoC), and defines rules for matching the points within those two groups so that each resulting point-pair consists of one point of the GoI and one of the GoC.

The signature of a comparative concept consists of a comparative dimension space. A comparative dimension space is an ordered set of two dimension spaces, whereby the first one defines the dimension space for the points in the GoI and the second one defines the dimension space for the points in the GoC. The interpretation of a comparative dimension space is the cross product of the points contained in its underlying dimension spaces.

The concept expression of a comparative concept may be given by (1) the combination of two MDCs and a join condition relating nodes of the GoI and the GoC by a conjunction of some predefined comparison operations, or (2) by a boolean expression over score-value-comparisons of scores applied to GoI and GoC [Neuböck et al., 2013, p. 17].

Example 9 (Comparative Concept and Comparative Dimension Space). The comparative concept with signature insDistrProv2012vs2011 (drugPrescriptionC) is based on the comparative dimension space drugPrescriptionC, which in turn has the signature drugPrescriptionC (drugPrescription [ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all], drugPrescription [ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all]). The concept expression of the comparative concept is defined by the two MDCs insDistrProv2012 and insDistrProv2011, whereby each of the two concepts is defined on the dimension space drugPrescription [ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all] with the interpretation of a restriction to districts and provinces in the dimension role ins at the specific node ttime:2012 and ttime:2011 respectively. Additionally, a join condition restricts the combination of the remaining points in the GoI and GoC so that only points that refer to the same node in the dimension role ins are combined. The resulting interpretation consists of each point in the GoI related to the point in the GoC that refers to the same node in the dimension role ins. All other dimension roles are restricted to their dimension-specific all-node. Figure 2.3 displays the different components of the comparative concept insDistrProv2012vs2011.

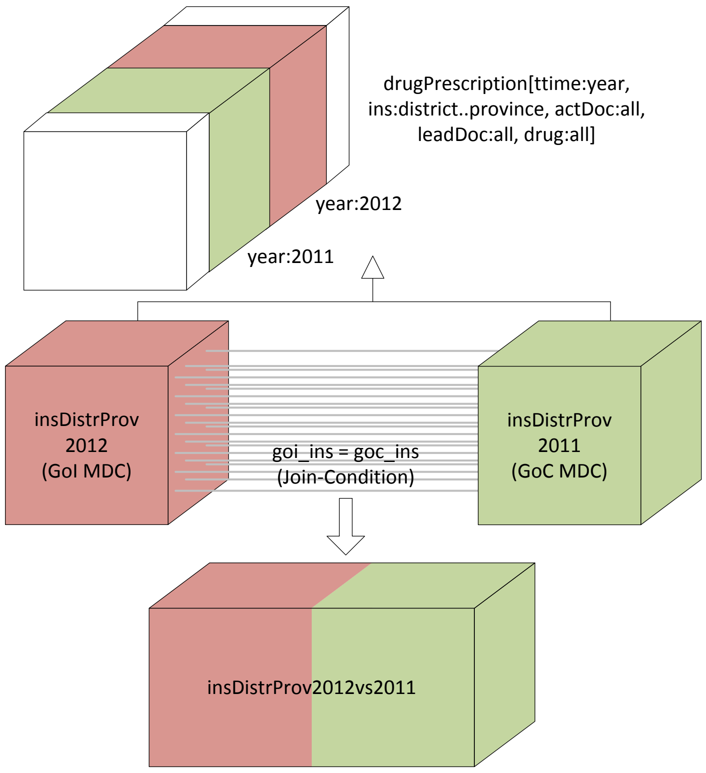
drugPrescription[ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all]
year:2012
year:2011
insDistrProv 2012 (GoI MDC)
goi_ins = goc_ins (Join-Condition)
insDistrProv 2011 (GoC MDC)
insDistrProv2012vs2011

Figure 2.3: Components of comparative concept insDistrProv2012vs2011

The semCockpit reasoning component applies reasoning to the concepts defined within the MDO. The reasoning process provides two specific results, which are subsequently used in other parts of the semCockpit system, namely subsumption hierarchy reasoning and reasoning over disjointness of MDO concepts.

Subsumption hierarchies facilitate query and concept definition. Through displaying the subsumption relations of MDO concepts analysts gain an overview on the available concepts and their general relationships. Therefore, it is easier to find and select similar concepts during query formulation. This is especially useful for coping with generic measures and scores, which will be introduced in section 2.4. Generic measures and scores define one or more generic qualifiers that are bound to a specific concept during their application. As the domain of generic qualifiers is usually defined by a subsuming concept, subsumption hierarchies can be used to display all concepts that can be used in place of a generic qualifier.

The reasoning over disjointness of concepts is used for the evaluation of judgement rules. Disjointness of two concepts asserts that two data cubes that are defined based on disjoint concepts do not contain any overlapping points. By determining disjointness of concepts, rules that do not correspond to a specific analysis situation can be eliminated, and therefore do not have to be evaluated. This elimination process uses the disjointness properties of the comparative concepts defined by the analysis situation and the cubes of judgement rules.

To derive knowledge about the relations of defined concepts, the web ontology language is used for reasoning. OWL provides automatic reasoning support for the generation of subsumption hierarchies and satisfiability checking. Neumayr et al. [2013] specify a language for defining multi-dimensional ontologies. In this approach, a multi-dimensional ontology is defined over the schema of a data cube and is interpreted by an extended data cube instance. Neumayr et al. define the representation of a data cube and MDO concepts in OWL. The semCockpit reasoning component uses this approach to generate OWL representations for the concepts defined in the MDO and delegates reasoning to an off-the-shelf OWL reasoner.

Reasoning of subsumption and disjoint relations is sound but incomplete in the prototype implementation. This is the result of a flexible concept definition approach, which allows the user to define concepts in many different ways, including concept definitions by providing a correct SQL representation of a concept and allowing to define concepts outside the MDO. Such concepts lack a corresponding OWL translation and are therefore not included in the reasoning process. Additionally, the reasoning might be incomplete for any concept that is defined on top of concepts without a complete OWL representation. The details of MDO representation and reasoning will be described in the Master's thesis of Christoph Ellinger, which is work in progress.

2.3 Measures and Scores

In this section a detailed introduction to measures and scores is provided. First, we provide the definition of MDO measures and explain the relationship between measures and other concepts and how concepts can be used for measure definition. In this context, we introduce the cube concept that is used within the MDO as means of defining measure applications. Then, we introduce scores as specific measure type for use in comparative data analysis and comparative cubes as means for defining comparative analysis situations.

In the context of measure and score expression definitions, MDO concepts are used as qualifiers to select the facts that are regarded for measure or score calculation.

Measures can be distinguished into base measures and derived measures. Base measures are derived from the fact classes of the semDWH and represent some recorded measure values in the context of business transactions. Derived measures are usually defined on base measures and represent some calculation or aggregation of base measure values. A derived measure might also be defined using other derived measures.

The signature of a measure consists of a dimension space. Base measures of fact classes are always defined on a monogranular dimension space, whereby each dimension role is restricted to the most fine-grained granularity level. The dimension space of a derived measure can be restricted freely. However, depending on the defined calculation of the measure and based on the dimension spaces of the measures the calculation is based on, implicit granularity restrictions apply. Disregarding these implicit restrictions can lead to the definition of a measure that does not contain any facts within its interpretation.

The definitions for base measures in the MDO are automatically derived from the fact classes defined in the semDWH. Base measures represent the set of measures that can be used as base for the calculation of other, derived, measures. As the signature of base measures is monogranular at the level of an individual transaction, base measures do not allow for multi-dimensional analysis. In order to support multi-dimensional analysis, derived measures are needed.

Derived measures are defined based on base measures or other derived measures. The calculation of a derived measure is specified by some mathematical term, which defines the calculation based on the underlying measure values. The dimension space of a derived measure can be multigranular. In order to create a multigranular measure from a monogranular base measure, the aggregation behaviour of the measure value has to be defined by an aggregation function. Typical aggregation functions are summation (SUM) and average (AVG).

Within the MDO, derived measure instructions can be defined in one of the following ways described in Neuböck et al. [2013, p. 23 ff.] (with different numbering): (1) by an arithmetic expression over measures, (2) by an aggregation expression over measures (defining one or more aggregation steps), (3) by a SQL expression over the semDWH, or (4) by instantiation of a generic measure.

Example 10 (Measure). The base measure costs as depicted in figure 2.2 contains a numeric value for the costs of a single drug prescription. Its dimension space is, therefore, for each dimension role restricted to the most fine-grained dimension level. If costs would be used in a multi-dimensional analysis that restricts the granularity of dimension role ins to level district and all other dimension roles to their respective all-node, the analysis would not yield a result as the base measure is not defined for this granularity. In order to make this analysis possible, a derived measure drugPrescriptionCosts can be defined. This measure might be defined over the unrestricted dimension space drugPrescription. In order to do so, the measure definition expression consists of the base measure costs as underlying measure and additionally defines the aggregation function SUM as function that is used to calculate values for roll-up points. The previously suggested analysis applied to this measure yields for each district in the dimension role ins the sum of the drug prescription costs over all drug prescription facts. By defining a measure qualifier it is possible to use a MDC in order to specify the points that are included in the measure computation. For example, the MDC InsurantInUrbanDistrict can be used as qualifier in the definition of the measure drugPrescriptionCosts. The definition of this qualifier denotes, that only facts for insurants that live in an urban district are added to the aggregated measure values.

A cube, also referred to as analysis situation, represents the application of one or more measures to a specific set of multi-dimensional points. A cube consists of a dimension space, a multi-dimensional concept and a set of measures. The dimension space of a cube represents the cube's signature. The multi-dimensional concept defines the set of points for which the measures should be evaluated. The set of measures comprises all the measures that are contained in the cube.

Example 11 (Cube). A cube representing the sum of drug prescription costs over districts and provinces for the year 2012 is defined by the multi-dimensional concept insDistrProv2012 (drugPrescription [ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all]) and the derived measure drugPrescriptionCosts (drugPrescription). The cube's signature can be derived from this definition as a restriction of the dimension space drugPrescription to the granularity range [ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all].

A score is used for defining the calculation of values in the context of a comparative analysis situation. A score value is calculated by relating measure values of two points, the point of interest (PoI) and the point of comparison (PoC). The signature of a score comprises of a comparative dimension space.

A score in the MDO is defined as (1) arithmetic score, (2) analytical score, or (3) as instantiation of a generic score [Neuböck et al., 2013, p. 26 ff.]. The score definition expression for an arithmetic score consists of two measures, one representing the measure for points in the group of interest (GoI), also referred to as goi-measure, and one for the points in the group of comparison (GoC), also referred to as goc-measure, together with an arithmetic expression, which defines the score value calculation based on the values of these measures. Typical expressions might define a score as *ratio* or *percentage difference* of goi-measure and goc-measure. Analytical scores define some analytical function, that is, *mean percentile rank* or *median percentile rank*, instead of an arithmetic function. Additionally, a score can be defined as instantiation of a generic score by specifying a valid concept for all generic score qualifiers of the generic score. Generic scores will be discussed in detail in section 2.4.

Example 12 (Score). In order to compare the drug prescription costs of different point groups, the score RatioOfDrugPrescCosts is defined. The score signature consists of the comparative dimension space drugPrescriptionC, which defines the dimension space drugPrescription as dimension space for the GoI as well as for the GoC. As the name suggests, the new score is based on the previously defined measure drugPrescriptionCosts for both GoI and GoC. Finally, the score calculation is defined as the ratio of goi-measure drugPrescriptionCosts and goc-measure drugPrescription-

Costs. The resulting score interpretation consists of a score value for each point-pair in the cross product of the points in the GoI and the GoC.

Comparative cubes (ccubes), also referred to as comparative analysis situations, define the application of scores to a set of comparative points. A comparative cube consists of a comparative dimension space, a comparative concept and a set of scores. The comparative dimension space represents the signature of the comparative cube. The comparative concept restricts the score calculation to the point pairs that are in the representation of this concept. Note, that the comparative concept does not influence the score calculation, but merely restricts it to a set of point-pairs. The set of scores contains the scores that should be evaluated for the specified point-pairs.

Example 13 (Comparative Cube). A comparative cube with the ratio of the sum of drug prescription costs over districts and provinces for the year 2012 compared to the year 2011, CostRatio2012, is defined by the comparative concept insDistrProv2012vs2011 with signature drugPrescriptionC (drugPrescription [ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all], drugPrescription [ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all]) and the score RatioOfDrugPrescCosts (drugPrescriptionC). The cube's signature can be derived from this definition as a restriction of the comparative dimension space drugPrescriptionC, (drugPrescription [ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all]), (drugPrescription [ttime:year, ins:district..province, actDoc:all, leadDoc:all, drug:all]). The defined comparative cube consists of a score value for RatioOfDrugPrescCosts for each comparative point within the interpretation of the comparative concept insDistrProv2012vs2011.

2.4 Generic Measures and Scores

This section introduces the concepts of generic measures and scores. First, we define the notion of generic MDO concepts as used in this work. We also introduce multi-dimensional metaconcepts, which are used for the definition of generic concepts. After that, the concept of generic measures is presented. We also discuss the necessary steps in order to instantiate a generic measure. Further, we provide the definition and instantiation process for generic scores. Finally, we introduce a generic extension of comparative cubes based on generic scores, which is needed for the definition of generic rules, that is, rules that are defined based on generic analysis situations.

Generic concepts in the MDO are generic in the sense that the concept expression is not fully defined at the time of the concept's definition. A generic concept is instantiated by qualifying the concepts that have been left open during definition of the generic concept. Only after instantiation of a generic concept a complete interpretation is possible. Therefore, generic concepts are only available within the MDO as it is not possible to translate them to semDWH objects.

Generic measures and scores are introduced in order to prevent the necessity to define similar yet different measures from scratch. A generic concept allows to define its calculation using generic qualifiers. A generic qualifier represents a variable for a concept that is only broadly restricted by the definition of the generic concept.

Instantiation of a generic concept is achieved by providing a concrete MDO concept for each of the generic qualifiers used in the generic concept definition. Therefore, given a generic concept and a set of qualifiers, a non-generic concept can be derived. The instantiation of a generic concept always results in a new non-generic concept of the same sort. Applying different qualifiers to a generic concept leads to different instantiated concepts. Therefore, multiple similar measures can be created by applying different qualifiers to a generic measure.

Within the definition of a generic concept, the domain of each of the generic qualifiers can be specified by a multi-dimensional metaconcept (md-metaconcept). Each generic qualifier can define a different domain with its own md-metaconcept. A md-metaconcept defines the concepts that may be applied to a generic qualifier during instantiation. If a concept that does not satisfy the defined md-metaconcept is applied, instantiation is not possible.

The definition of a md-metaconcept can be given in two ways; either by subsumption of a specific concept or by a set of concepts. The interpretation of a md-metaconcept by subsumption is as follows: each concept that is subsumed by the specified concept, as well as the specified concept itself, is in the interpretation of the md-metaconcept and is therefore a valid concept during instantiation. In the case of a given set of concepts, only the exact concepts that are contained in the set are valid qualifiers for instantiation.

The signature of a generic measure consists of a dimension space and a set of generic qualifier names used in the measure definition expression. The set of qualifiers is needed so that they can be referenced for the instantiation of the generic measure. The domain of each generic qualifier in a generic measure definition has to be restricted by a md-metaconcept.

The concept expression of a generic measure can be given (1) by an arithmetic expression over measures or (2) by an aggregation expression over measures [Neuböck et al., 2013, p. 27]. These expressions are analogous to the definition expressions of non-generic measures with the exception that generic qualifiers can be used in the definition expression.

The definition of a generic measure can be based on other generic measures. Such measures are called higher-level generic measures. In the case of a higher-level generic measure the new generic measure has to bind each of the generic qualifiers of the underlying generic measure to either a concrete concept within the interpretation

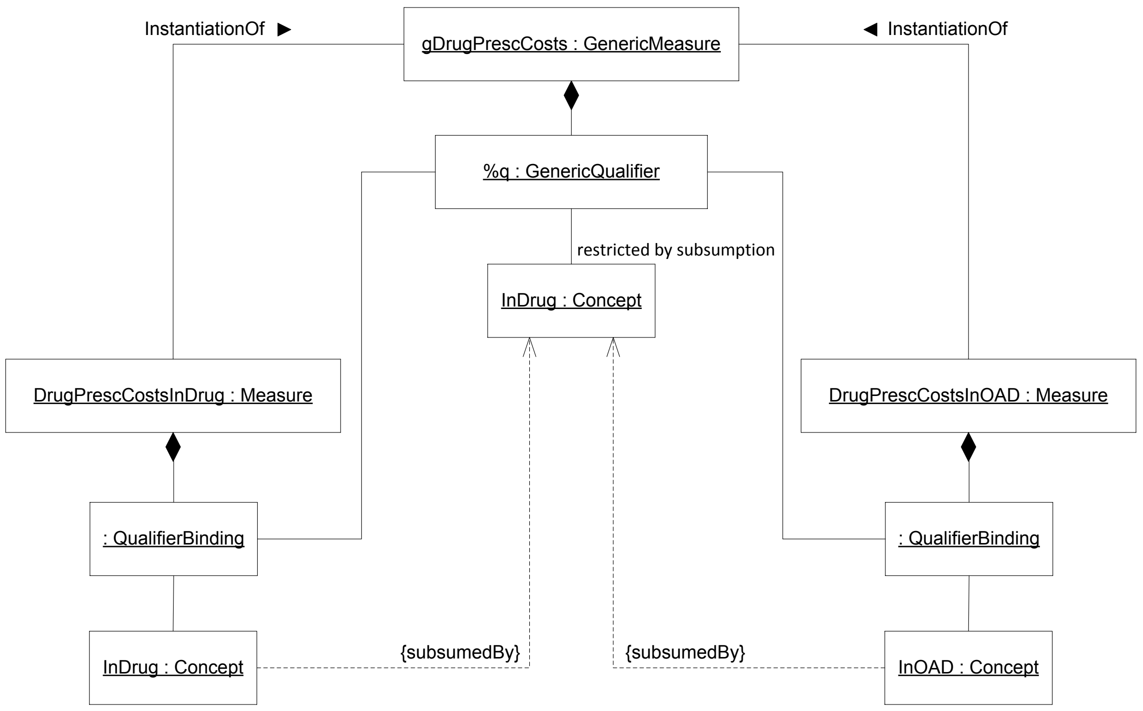

Figure 2.4: Relationship of generic measure gDrugPrescCosts and its instantiations DrugPrescCostsInDrug and DrugPrescCostsInOAD

of the corresponding md-metaconcept (static qualifier binding), or to one of the generic qualifiers of the new measure (dynamic qualifier binding). By using multiple qualifier bindings it is also possible to bind two or more different generic qualifiers of the underlying generic measure to the same generic qualifier in the higher-level measure expression. Instantiation of a higher-level generic measure leads also to an instantiation of each of the generic measures used in the measure definition expression of the higher-level generic measure.

Example 14 (Generic Measure). Figure 2.4 depicts the relationship of the generic measure gDrugPrescCosts and two instantiations DrugPrescCostsInDrug and DrugPrescCostsInOAD. The expression definition of gDrugPrescCosts is defined as the sum of all drug prescription costs that involve the drug specified by the generic qualifier %q. The domain of the generic qualifier is specified as concepts that are subsumed by the multi-dimensional concept InDrug, which has the interpretation of any node contained in the dimension role drug. Measure DrugPrescCostsInDrug instantiates the generic measure by binding the concrete concept InDrug to the generic measure qualifier %q, resulting in a measure with the interpretation of the sum of all drug prescription costs. Measure DrugPrescCostsInOAD instantiates the same generic measure by binding the concept InOAD to the generic qualifier %q. The concept InOAD is interpreted as containing all points of the dimension role drug that can be subsumed by the node oral anti-diabetic drugs. InOAD is subsumed by concept InDrug and is therefore a valid binding for the measure instantiation. The resulting measure DrugPrescCostsInOAD is interpreted as the sum of all drug prescription costs for facts that involve some oral anti-diabetic drug.

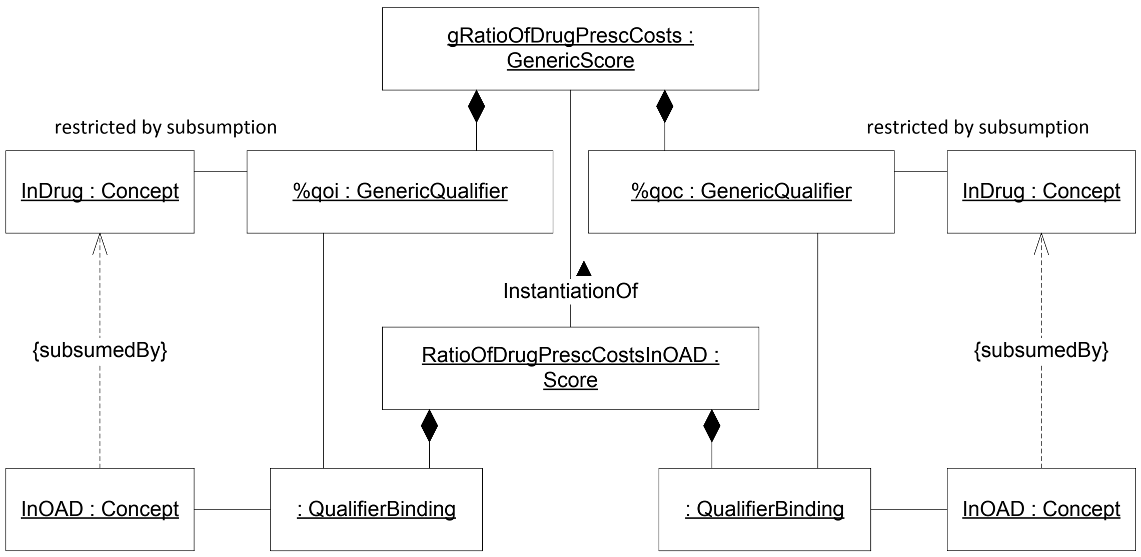

Figure 2.5: Relationship of generic score gRatioOfDrugPrescCosts and the instantiated score RatioOfDrugPrescCostsInOAD

Generic scores are score definitions that use generic qualifiers instead of concrete concepts in the score expression definition. Like generic measures, the domain of each generic qualifier of a generic score has to be restricted by a md-metaconcept.

If the definition of a generic score uses generic measures, then the binding of the generic qualifiers and concepts to the qualifiers of the used generic measures has to be defined, as in the case of a higher-level generic measure.

Example 15 (Generic Score). Figure 2.5 shows the qualifier bindings of the score RatioOfDrugPrescCostsInOAD defined as an instantiation of the generic score gRatioOfDrugPrescCosts. The generic score is defined as the ratio of the generic measure gDrugPrescCosts for the GoI and the same generic measure for the GoC. The generic score defines two variable generic qualifiers (%qoi and %qoc), which are bound to the variable generic qualifiers of the underlying goi-measure (%q) and goc-measure (%q). The instantiation of the generic score, therefore, has to define a concept for each of the two generic qualifiers. Depending on the used qualifiers the resulting score has a different interpretation. In the case illustrated in figure 2.5 the concept InOAD is used for both generic qualifiers. As the score qualifiers are bound to the underlying generic measures, the resulting score can also be defined as the ratio of measure DrugPrescCostsInOAD and DrugPrescCostsInOAD with an interpretation as the ratio of drug prescription costs that involve some node from dimension role drug that is in the interpretation of the concept InOAD, calculated for the cross product of the points in the GoI and GoC.

The concept of a generic comparative cube is used in order to define a generic comparative analysis situation, which in turn can define the scope of rules (see section 3.3). A generic comparative cube is a comparative cube that is defined using one or more generic scores.

The signature of a generic comparative cube is defined by a comparative dimension space. The concept definition expression consists of (1) a comparative concept, (2) a set of generic scores and (3) a set of md-metaconcepts, which represent the qualifier domains of the generic qualifiers defined by the generic scores of the cube.

A generic comparative cube can define additional restrictions to the domain of the generic qualifiers used in the definition of its generic scores. For each generic qualifier of a generic score encompassed by the generic cube definition a domain can be specified using a md-metaconcept. This md-metaconcept can further restrict the domain of the generic qualifier as specified in the generic score definition. This circumstance allows to define comparative cubes that are restricted to the defined domain in the sense that rules for their facts are only evaluated if each concrete qualifier is valid with respect to the specified cube domain of the generic qualifier it is bound to. In order to enable a valid instantiation the restriction specified in the generic cube definition has to be consistent with the restriction imposed by the md-metaconcept of the generic score qualifier.

Example 16 (Generic Comparative Cube). In the previous section, the comparative cube CostRatio2012 has been defined containing the ratio of the sum of drug prescription costs over districts and provinces for the year 2012 compared to 2011 over all drug prescription transactions. A similar analysis might want to gauge this ratio for a specific type of drugs. In order to support this demand without having to define a comparative cube for each drug type a priori, a generic comparative cube can be defined. This generic comparative cube is defined using the generic score gRatioOfDrugPrescCosts. As the points that should be compared are the same as for the non-generic comparative cube CostRatio2012, the same comparative concept insDistrProv2012vs2011 is used for the definition of the generic cube. A generic comparative cube allows to further restrict the qualifier domain of the generic scores underlying its definition. For example, the qualifier domain of the generic qualifier %qoi of the generic score gRatioOfDrugPrescCosts is defined by subsumption of the concept InDrug. This domain could be further restricted, for example, to a set of specific drug type concepts like InOAD. Instantiation of a generic comparative cube is only valid, if the concrete qualifiers obey the most restricted domain definition for the bound generic qualifiers.

2.5 MDO-DWH Mapper

The MDO-DWH Mapper is responsible for the communication between the semCockpit data warehouse and the multi-dimensional ontology. This includes the execution of statements derived from MDO construct definitions in the semDWH and the loading of information from the semDWH into the MDO database. Retrieving information from the semDWH is primarily used during the initialisation process in order to model the DWH structure in terms of MDO constructs. Execution of SQL statements in the semDWH is used to create MDO concepts as objects in the semDWH.

Introduction of an independent mapping component allows a loose coupling between semDWH and MDO-DB. Therefore, technologies used for the implementation of semDWH and MDO do not strictly depend on each other as long as the Mapper is able to handle the necessary communication methods. This has been an explicit consideration during the conceptualisation of the prototype implementation, as this allows to replace the technology used for implementation of one component without heavily effecting other components. Therefore, subsequent research can be conducted on the suitability of different technologies for different semCockpit components.

The most important function of the Mapper is to provide methods for translating MDO concept definitions to data definition language (DDL) statements. The language definition for translating MDO concepts to SQL is specified in Neumayr et al. [2013]. The mapping component provides an interface that allows to control the mapping process, that is, executing the generated statements in the semDWH and loading information from the semDWH into the MDO-DB.

The execution of the generated statements in the semDWH transfers MDO concepts to the data warehouse so that the concepts can be used in the semDWH. As soon as the DDL statements for the creation of a specific concept have been executed in the semDWH, the concept is available for DWH queries.

Due to the intended loose coupling of MDO and semDWH, and different character constraints for identifiers in different technologies, the Mapper also needs to maintain a name registry. The name registry maps the name of a concept in the MDO to the semDWH object that represents the concept in the data warehouse and vice versa. Therefore, the naming of MDO concepts does not necessarily have to abide by technology-specific constraints such as the 30 character length limit for construct identifiers in Oracle database systems.

It is important to execute the SQL statements in the correct order in the semDWH as MDO constructs might be defined based on other MDO constructs. To ensure that a statement can be correctly executed in the semDWH it is necessary that the statements for each underlying construct have already been executed. The Mapper ensures the correct execution order by assigning an incremental sequence number to each mapping command, which defines the order of the statement execution in the semDWH.

The MDO-DWH Mapper also provides command-pattern like functionality for undoing and redoing previous mapping steps. This allows, for example, to reconstruct the state of the semDWH at a previous point in time. In order to do so the Mapper uses the sequenced list of commands that have to be executed in the semDWH. Each command defines a corresponding statement that constitutes the nullification of the initial command. An additional variable keeps track of the last statement that has been executed in the semDWH. The Mapper allows to move along this command sequence in order to recreate previous states of the semDWH.

If a construct is deleted from the semDWH, then all constructs that directly or indirectly contain this construct in their definition are invalidated and are no longer executable in the semDWH.

3 Judgement and Analysis Rules

In this chapter the basic concepts that underlie the semCockpit rule engine are described. We explain the characteristics and behaviour of rules in the context of the semCockpit system. This includes the definition of judgement and analysis rules. As extension to these types we provide the definition of generic judgement and analysis rules as rules that are defined on generic comparative cubes representing generic analysis situations. This will lead to an understanding of the fundamental structure of the prototype implementation, which will be discussed in detail in chapter 4.

As mentioned previously, the rule engine implementation described in this thesis is built on top of the semCockpit data warehouse (semDWH) and its multi-dimensional ontology database (MDO-DB).

In OLTP database systems the Event-Condition-Action (ECA) model introduced by McCarthy and Dayal [1989] is used for modelling business rules. An event is the trigger of a rule in the sense that it causes the system to evaluate the rule condition. The condition of a rule, according to McCarthy and Dayal [1989, p. 216], is defined as a set of queries, which is evaluated when an event triggers the rule. If an event triggers a rule and its condition is satisfied, then the action is executed. Analogous, the ECA model can also be applied to data warehouse systems. Thalhammer et al. [2001] specifically discuss the realisation of ECA rules in the context of active data warehousing. Thalhammer et al. established the notion of analysis rules as data warehouse specific rules.

Research in the field of active data warehousing focuses primarily on two complementary aspects, the ETL process and the analytical aspect. In order to fully utilise the reactive capabilities of an active data warehouse (ADW) it is necessary to ensure that the underlying data is updated in a timely fashion. Therefore, the ETL process received considerable attention, which lead to the notion of right-time and real-time data warehouses [Araque, 2003; Mohania and Narang, 2003; Golfarelli et al., 2004]. The second option is to approach active data warehousing from an analytical point of view and focus on the reactive decision system capable to perform automatic analytical tasks. According to Thalhammer et al. [2001, p. 267] analysis rules provide a mechanism to analyse data multi-dimensionally and to make decisions based on analysis results. Analysis rules use the ECA model and extend it with data warehouse specific functionality for coping with the inherent multidimensionality of data warehouses. The event of an analysis rule specifies the time points at which the rule should be evaluated; the condition is a boolean predicate or a query, which leads to execution of the specified action; the action is a directive to execute a transaction for some entities in the OLTP database. Based on this proposal several implementations of ADWs have been proposed. Thalhammer and Schrefl [2002] show how analysis rules can be implemented on top of off-the-shelf database technology. Zwick et al. [2007] use workflow engines in order to implement automated analyses in active data warehouse systems. They implement the decision making process of data analysis based on analysis graphs consisting of three primitives; AnalysisStep, AnalysisLoop, and Action. Bouattour et al. [2009] define a new formalism of analysis rules and a general framework of an active data warehouse. Their approach focuses on the use of XML to model the logical and physical level of analysis rules.

Neumayr et al. [2011] first introduced judgement rules and guidance rules for supporting business analysts in comparative data analysis. Judgement rules are used to annotate domain specific knowledge in order to explain extraordinary score values for a set of comparative facts. Guidance rules operate on BI analysis graphs [Neuböck et al., 2012]. A BI analysis graph describes promising navigation steps for a specific analysis situation. Therefore, a BI analysis graph provides some guidance for the business analyst by suggesting promising OLAP-steps. Guidance rules are used to augment BI analysis graphs by enabling the definition of conditions under which a specific analysis step is suggested [Neuböck et al., 2013, p. 37f.]. Based on the concept of analysis rules, Neuböck et al. [2013] identified various variations of rules that can be implemented and used in a data warehouse environment, including reporting rules and action rules [Neuböck et al., 2013, pp. 37-42]. Reporting rules report, based on defined evaluation strategies, the result of a multi-dimensional analysis to the business analyst. Action rules are similar to reporting rules in the sense that they apply multi-dimensional analysis. However, instead of reporting the analysis results to the analyst, some predefined actions are executed. Due to their similarity, Neuböck et al. [2013] also use the term analysis rule interchangeably for both reporting and action rules.

Current business intelligence solutions provide some support for rule functionality in terms of defining thresholds and reporting threshold exceedings [Browne et al., 2010; Greenwald et al., 2007]. For example, current Oracle databases support a feature called Delivers by defining alerts that trigger based on user-specified conditions and lead to, for example, an email notification to the analyst [Greenwald et al., 2007, p. 242]. As a different example, IBM Cognos supports a sophisticated comment function, which can be used to annotate judgements to specific reports and report parts [Browne et al., 2010, p. 212]. Note, however, that this functionality is different to judgement rules in that annotations for each cell have to be created and maintained by the user and are not automatically generated based on a rule definition.

Our work draws ideas from active data warehouses, especially from analysis rules [Thalhammer et al., 2001] and focuses on the analytical aspect of data analysis. We enable to define rules for comparative analysis situations and to raise judgements and reports in order to initiate further user interactions. With the prototype implementation, we augment the ontology-based business intelligence approach of the semCockpit system with rules that provide some active data warehouse functionality. As part of the semCockpit approach explicitly defined concepts describing business terms can be used for the definition of rules.

This work focuses on rules for comparative is-to-is analysis situations. However, the principles described here are also applicable for is-to-target comparison, as well as for non-comparative analysis situations. This focus is motivated by the limited support for comparative data analysis in current BI tools. Additionally, if the presented approach can be implemented for comparative is-to-is analysis, then it can be deduced that an implementation for other, less complex, analysis situations based on the same principles is also feasible.

The prototype supports two different types of rules, namely judgement and analysis rules. Analysis rules in this work are not strictly divided into reporting and action rules. As our prototype lacks an action execution model arguably only reporting rules are really implemented (see chapter 4). The necessary steps in order to fully implement the functionality of action rules are discussed in section 4.7.

In addition to a distinction based on the type of a rule, rules are also distinguished into generic and non-generic rules. In the remainder of this work, non-generic rules are also referred to as base rules. Generic judgement and generic analysis rules define the same features as their non-generic counterparts with the exception that a generic rule is defined based on a generic comparative analysis situation instead of a non-generic one. Further details with respect to generic rules are provided in section 3.3.

Definitions of rules in the semCockpit system are also part of the MDO. Like other MDO concepts, rule definitions are stored in the repository in order to explicate them. The entirety of all business rule definitions in the MDO is defined as the semCockpit rulebase.

Each rule belongs to a rule family and defines a generic or non-generic comparative cube as well as a rule condition. Additional characteristics depend on the specific rule type.

The scope of a rule is defined by a comparative cube. This cube qualifies the set of comparative facts for which the rule applies. Comparative cubes can be arranged in subsumption hierarchies based on their respective sets of comparative facts. The comparative points contained in a comparative cube are defined by a comparative concept. Therefore, the hierarchy of comparative concepts can be used in order to

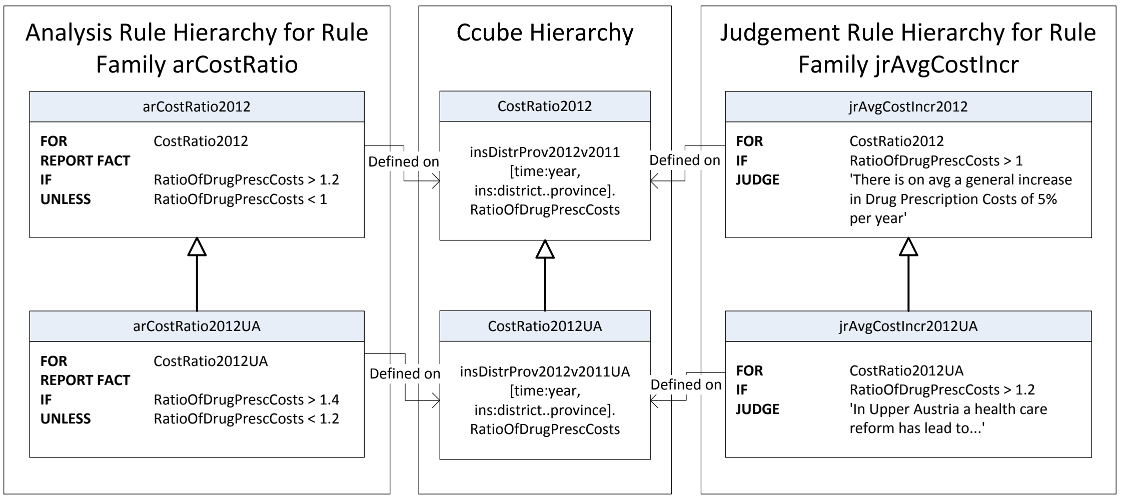
Analysis Rule Hierarchy for Rule Family arCostRatio
arCostRatio2012
FOR CostRatio2012
REPORT FACT
IF RatioOfDrugPrescCosts > 1.2
UNLESS RatioOfDrugPrescCosts < 1
arCostRatio2012UA
FOR CostRatio2012UA
REPORT FACT
IF RatioOfDrugPrescCosts > 1.4
UNLESS RatioOfDrugPrescCosts < 1.2
Defined on
Ccube Hierarchy
CostRatio2012
insDistrProv2012v2011 [time:year, ins:district..province]. RatioOfDrugPrescCosts
CostRatio2012UA
insDistrProv2012v2011UA [time:year, ins:district..province]. RatioOfDrugPrescCosts
Judgement Rule Hierarchy for Rule Family jrAvgCostIncr
jrAvgCostIncr2012
FOR CostRatio2012
IF RatioOfDrugPrescCosts > 1
JUDGE 'There is on avg a general increase in Drug Prescription Costs of 5% per year'
jrAvgCostIncr2012UA
FOR CostRatio2012UA
IF RatioOfDrugPrescCosts > 1.2
JUDGE 'In Upper Austria a health care reform has lead to...'

Figure 3.1: Analysis and judgement rules

decide the hierarchical order of comparative cubes as long as they are defined on the same scores. Again, the details of the reasoning process are described in the Master's thesis of Christoph Ellinger, which is work in progress. In this work, we simply assume the existence of a reasoning component that provides sound reasoning results. The scopes of multiple rules belonging to different rule families can be defined by a single comparative cube.

Example 17 (Comparative Cube Hierarchy). Figure 3.1 shows the two comparative cubes CostRatio2012 and CostRatio2012UA in the same cube hierarchy with the cube CostRatio2012UA lying below cube CostRatio2012 in the cube hierarchy. Cube CostRatio2012UA contains a subset of the comparative facts contained in cube CostRatio2012, namely only those facts that are contained in the interpretation of the concept InUpperAustria. The comparative cubes can be used for the definition of rule scopes. The hierarchy is derived from the subsumption relation of the comparative concepts insDistrProv2012v2011 and insDistrProv2012v2011UA, which specify the comparative points contained in the example cubes. The comparative cube CostRatio2012 specifies the scope of the two rules arCostRatio2012 and jrAvgCostIncr2012. The cube CostRatio2012UA represents the scope for the analysis rule arCostRatio2012UA and the judgement rule jrAvgCostIncr2012UA.

The condition part of rules consists of score-value-comparisons for non-generic rules and generic score-value-comparisons for generic rules (see figure 3.2). Conditions can only involve scores defined by the rule's comparative cube. Several score-value-comparisons can be concatenated by boolean expressions AND and OR, in order to form more complex conditions. If the condition of a rule is satisfied by a comparative fact then the rule fires and executes the rule-specific action. The detailed action depends on the type of the rule, its definition, and in the case of

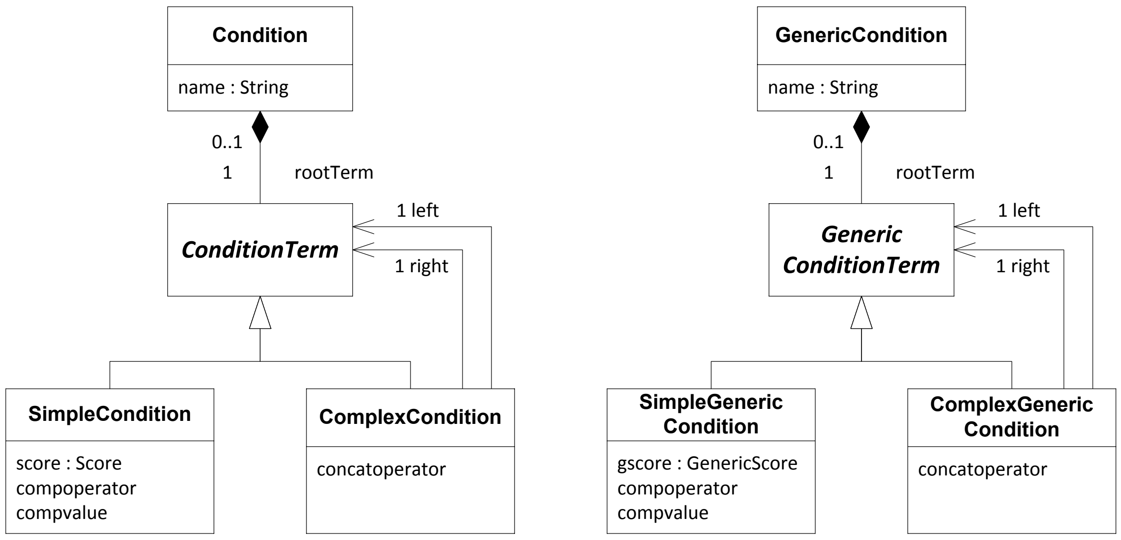

Figure 3.2: Conceptual model of non-generic (left) and generic (right) rule conditions

analysis rules on the selected evaluation strategy. Details of the different actions are discussed in the following sections with regard to each type.

In order to support the definition of context-specific rules, rules are arranged in rule families. Each rule family consists of one or more rules sharing the same type. Three different rule family types are distinguished: (1) base rule family, (2) generic rule family, and (3) rule family by generic instantiation. A base rule family consists of non-generic rules, whereas a generic rule family consists of generic rules. A rule family by generic instantiation is a special non-generic rule family type that represents an instantiation of a generic rule family. Note, that rules can only be assigned to base rule families (non-generic rules) or generic rule families (generic rules). The rules that belong to a rule family by generic instantiation are derived from the generic rule family that is instantiated. Rules within a rule family have different scopes, which determine their context. It is not possible to define two rules for the same set of comparative facts within the same rule family. Rule conditions are defined for each rule individually.

Rules within a rule family are hierarchically ordered from most general to most specific. This rule hierarchy can be derived from the hierarchy relations of the comparative cubes that underlie the rule definitions. The reasoning strategy for rule hierarchies is part of the ontology reasoner component, and therefore not covered by this thesis.

Example 18 (Rule Hierarchy). The judgement rule family jrAvgCostIncr, depicted in figure 3.1, consists of two rules, one defined on cube CostRatio2012 and one defined on cube CostRatio2012UA. The rule hierarchy for jrAvgCostIncr can be derived from the hierarchy of the relevant cubes. As cube CostRatio2012UA is below cube CostRatio2012 in the cube hierarchy it is derived that rule jrAvgCostIncr2012UA is below rule jrAvgCostIncr2012 in the rule hierarchy of the judgement rule family jrAvgCostIncr. Analogously, the rule hierarchy for analysis rule family arCostRatio can be derived. Therefore, analysis rule arCostRatio2012 is the root rule of the rule family with rule arCostRatio2012UA being below rule arCostRatio2012 in the rule hierarchy.

In order to allow a defined rule evaluation it is assumed that rule definitions define a consistent hierarchy in the sense that each hierarchy has a single root rule that subsumes all other rules that are part of the rule family.

Additionally, some constraints concerning overlaps in rule hierarchies are necessary in order to ensure a defined evaluation of rule families. Without further constraints the scopes of rules in sub-hierarchies of a single rule family may overlap. Therefore, two comparative cubes in different sub-hierarchies of a rule family may have some comparative points in common. Such overlaps can lead to unexpected behaviour during rule evaluation of analysis rules as it is not possible to obtain a single most specific rule for evaluation. In order to prevent unintended behaviour, some validation is needed that assures that the sets of comparative points in different sub-hierarchies of an analysis rule family are disjoint. This validation process is not implemented in the current prototype implementation of the rule engine. If the necessity arises, the constraint can be implemented as part of the user interface input validation. Note, that this restriction only applies to analysis rules. For judgement rules overlaps in the rule scopes do not prevent a defined rule evaluation, however, the concrete behaviour has to be defined. A detailed description of the evaluation behaviour will be discussed with regard to the specific rule types in the subsequent sections.

The application of rules is based on rule families. A single rule within a rule family is generally only evaluated in the context of its rule family. During rule family evaluation, for each fact the most specific rule within the family's rule-hierarchy is applied. If a comparative point is within the scope of two different hierarchies of a judgement rule family, then the most specific rule of both hierarchies is evaluated for this point.

In the following sections the two different base types of rules, analysis rules and judgement rules, that are part of the prototype implementation together with the corresponding base rule families are discussed in detail. Following this, a description of generic judgement and analysis rules is provided.

3.1 Judgement Rules

This section presents the concept of judgement rules as implemented by the rule engine. First, we provide the general idea of judgement rules and their motivation. Then, we give the definition of judgement rules and judgement rule families. Finally, we discuss the evaluation process for judgement rules.

Judgement rules can be used to annotate certain groups of facts with specific information and knowledge, and therefore express some, otherwise tacit, knowledge about the underlying data [Neuböck et al., 2013, p. 3]. For example, a judgement rule might inform the analyst about the average yearly increase in drug prescription costs for comparative analysis situations that compare drug prescription costs of the current year to the costs of the previous year.

The textual judgement of a rule is displayed for a comparative fact when its score values satisfy the condition of the rule. Most state-of-the-art business intelligence tools support a comparable mechanism with alerters for thresholds [Browne et al., 2010; Greenwald et al., 2007]. Judgement rules allow more sophisticated definitions as they are naturally defined over comparative analysis situations. Further, they allow the usage of MDO business terms for rule definition. Judgement rules are also explicitly defined and stored in the MDO repository, allowing for an overview of the currently defined rules at every time.

A judgement rule is defined by (1) a comparative cube, (2) a condition, and (3) a corresponding judgement and (4) belongs to exactly one judgement rule family [Neuböck et al., 2013, p. 38]. A rule's comparative cube defines the comparative facts that are in the scope of the rule. The rule condition defines the thresholds that lead to a judgement for facts within the rule scope. The judgement is a useful information or annotation in textual form, which is displayed for facts that satisfy the rule condition. The rule family defines the context of the rule. Rules belonging to the same rule family override each other so that for each comparative point only the judgements of the most specific rules are contained in the evaluation result.

A judgement rule family consists of a set of judgement rules organised in a rule hierarchy defining a single root rule. Additionally, a judgement rule family defines a set of scores that contains all scores that are used by at least one of the comparative cubes of the rules that belong to the rule family.

Figure 3.3 shows the conceptual structure of judgement rules. Note, that only base judgement rule families and the corresponding non-generic rules are described in this section. For a detailed description of generic judgement rule families and judgement rule families by generic instantiation refer to section 3.3.

Evaluation of judgement rules is initiated implicitly for all rule families that match a given analysis situation. Depending on the concrete implementation, judgement rule evaluation might be activated by default and deactivated on demand for single

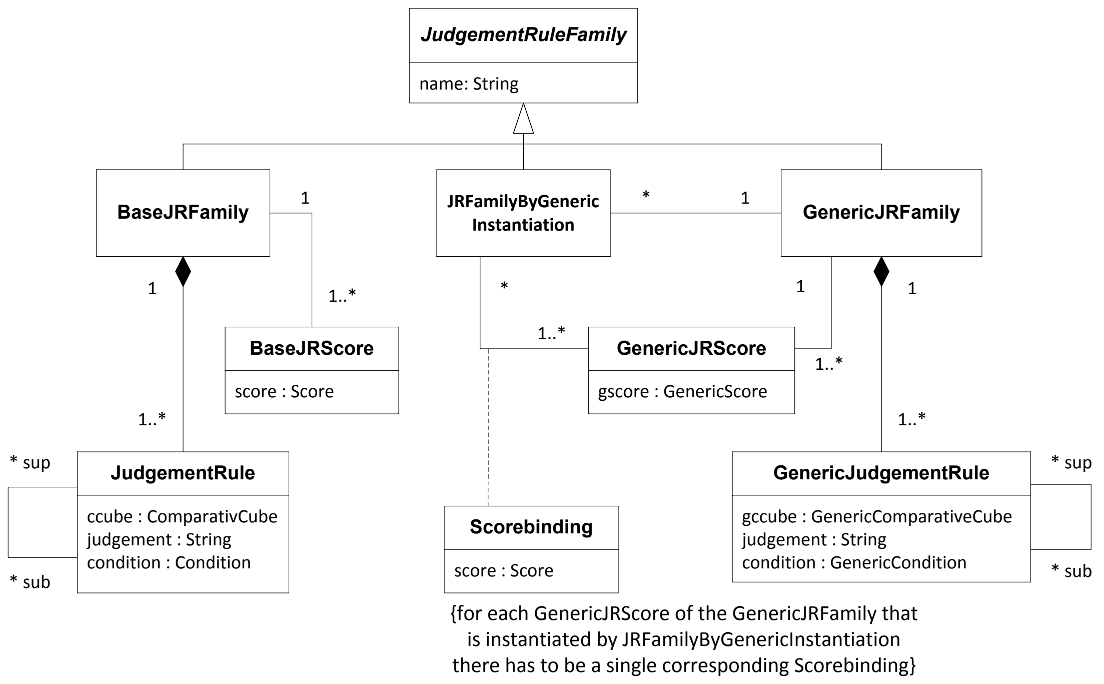

Figure 3.3: Conceptual model of judgement rules and judgement rule families

analysis situations or vice versa. Such an analysis situation is always specified as non-generic comparative cube. Each time an analysis is executed, all judgement rule families defined for facts contained in the analysis situation are evaluated. For those facts that satisfy the condition of the corresponding rule the specific judgement is displayed together with the results of the initial analysis.

In terms of the ECA model a judgement rule family might be described as follows: The event of a judgement rule family is the execution of an analysis situation that contains all of the scores over which the rule family is defined and has potentially overlapping comparative points. The process of determining such potential overlaps will be discussed later in this section. The condition part is a boolean expression that has to hold in order to activate the rule's action. Finally, the action triggered by a judgement rule consists, for each triggering fact, of the additional report of the specified textual judgement in the analysis result.

Example 19 (Judgement Rule). Judgement rule family jrAvgCostIncr consists of the two judgement rules jrAvgCostIncr2012 and jrAvgCostIncr2012UA, with rule jrAvg-CostIncr2012 being the more general, and therefore the root rule of the rule family hierarchy. The rule family is defined over the score RatioOfDrugPrescCosts. Judgement rule jrAvgCostIncr2012 defines a general judgement for the average drug cost increase from year 2011 to 2012 for districts and provinces of the dimension role ins. The more specific rule jrAvgCostIncr2012UA defines a more detailed judgement for the increase in drug prescription costs in the province of Upper Austria and its districts due to a health care reform. As the two rules belong to the same rule family, only the most specific rule is evaluated for each comparative point. A comparative point contained in the comparative cube CostRatio2012UA can either lead to the judgement defined by jrAvgCostIncr2012UA if the corresponding value of RatioOfDrugPrescCosts is above the defined threshold of 1.2, or no judgement at all. Even if such a point would have a corresponding score value of 1.1, which is above the threshold defined by the general rule jrAvgCostIncr2012, no judgement is provided, as the general rule is overridden by the more specific rule jrAvgCostIncr2012UA.

A rule applies to an analysis situation if the facts of the comparative cube that defines its scope overlap with the comparative cube of the analysis situation. Therefore, the scores of the rule's comparative cube have to appear in the analysis situation and additionally some comparative points have to appear in both the analysis situation and the comparative cube of the rule.

For deciding whether the facts of a rule family and a given analysis situation overlap, both the scores defined by the rule family and the comparative cube of its root rule are used. First, the scores of the analysis situation are compared to the scores of the rule family. Second, we use the comparative concepts of the comparative cubes of both the analysis situation and the root rule in order to decide whether some comparative points of the analysis situation are within the scope of the defined analysis rule family. Combining the results of these comparisons, rule families that do not contain overlaps can be detected and subsequently excluded from further judgement rule evaluation.

During evaluation, rule families are filtered for families defined over the same scores, or a subset thereof, as contained in the analysis situation. Each score of a rule family has to be part of the analysis situation for the rule family to be evaluated. If one or more of the scores of a rule family do not appear in the analysis situation, then the rule family is not considered applicable for this analysis situation and can be excluded from evaluation. Further specifics on the score matching for generic judgement rule families are discussed in section 3.3.

In the context of rule families it is sufficient to check the most general, or root rule, of a rule family for overlapping comparative points with the analysis situation. If some overlaps are detected for the root rule, the rule family is applicable to the analysis situation, and therefore evaluated. On the other hand, if an overlap can be ruled out, the corresponding rule family can be ruled out as well, as all other rules in the family, according to the defined properties of rule hierarchies, can only contain a subset of the comparative points of the root rule, and therefore cannot overlap either.

After the selection of relevant judgement rule families, all potentially relevant rules left are evaluated for the given analysis situation by adding judgements based on the rule definitions. Each comparative fact of the initial analysis situation is part of the evaluation result, whether a judgement is available or not. This reflects the concept of judgements as additional information, which analysts can use in addition to conventional analysis results. For each fact for which a judgement is available, its judgement and the specific rule that triggered the judgement is reported together with the conventional analysis result for the fact. A fact in the evaluation result can have multiple judgements from different rule families. Multiple judgements are represented by multiple instances of the comparative fact in the evaluation result, one for each available judgement and its triggering rule.

For judgement rules an easy and consistent way to handle overlaps in rule hierarchies can be defined by evaluating both rules for the concerned points. Therefore, it is not necessary to constrain judgement rule hierarchies in order to avoid overlapping sub-hierarchies. Three different cases can occur during evaluation when two rules in different branches of the same judgement rule family overlap in the sense that they are defined for the same comparative point. If the condition for neither rule applies, then there is no reported judgement for the comparative fact. Conversely, if the condition evaluates to true for both rules, then the judgement of both rules is reported in the same way as if two rules of different rule families fire. If only the condition of one of the rules applies, then only the judgement of that rule is reported.

Exclusion of not applicable rule families might be incomplete, leading to the evaluation of rules that do not apply to the evaluated analysis situation. The design of a potentially incomplete exclusion function is based on the following considerations: The MDO is designed to support the definition of concepts in many different ways. This includes, for example, the possibility to directly define a concept in the MDO by providing the concept's SQL implementation in the semDWH. As the reasoning component only has access to the definitions in the MDO it cannot compute the corresponding OWL representation of such a concept. This leads to an incomplete reasoning result. As the results provided by the reasoning component might be incomplete it is not possible to implement a complete exclusion function for the filtering of relevant rule families.

The described exclusion function leads to the same evaluation result no matter if the reasoning is complete or not, as long as the reasoned relations are sound. Evaluation of rules that in fact do not overlap with facts of the analysis situation do not lead to any additional judgements. The evaluation of such rules only raises the computational cost during rule evaluation. On the other hand, removing a rule family from evaluation even though overlapping of facts could not be eliminated, can lead to a different evaluation result, as the excluded rule family might contain some relevant judgements for the analysis situation in question. In order to guarantee correct evaluation results, the described exclusion approach has been defined.

This exclusion approach does not apply to the matching of scores. Conceptually, the analyst is only interested in judgements that are based on one or more scores of the current analysis situation. Evaluation of rule families that define different scores than the analysis situation but do have overlapping comparative points might lead to irrelevant and distracting judgements and is therefore avoided.

For the matching of scores no reasoning support that can be used to decide whether two scores that are defined differently are equal, exists. As the MDO allows to define scores in several different ways and no canonical form of score definition exists, it is difficult to compare and match scores that intrinsically lead to the same results if they have been defined in different ways. The discussed prototype does not support sophisticated equality-checks for scores as it is assumed that such checks are implemented within the MDO. For simplicity it is assumed that score definitions within the MDO are unique in the sense that two scores are equal only if they share the same name. Therefore, the prototype would treat two scores that are defined in the same way but do not share the same name as different, non-matching, scores for the selection of relevant rule families.

3.2 Analysis Rules

In this section we discuss the concept of analysis rules. First, we provide the general idea of analysis rules in the context of the rule engine prototype and provide a definition of the components of an analysis rule. Further, we explain the evaluation of analysis rules and discuss two different evaluation strategies, prerogative and presumed evaluation.

Analysis rules offer a method for analysing specific data in a hierarchical order and result in recommended actions (action rules) or an analysis report (reporting rules). Analysis rules can be used to model routine and semi-routine decision tasks in order to support business analysts [Neuböck et al., 2013, p. 37].

An analysis rule is defined by (1) a comparative cube; (2) two conditions, one representing positive activation (IF) and one representing negative activation (UNLESS); and (3) belongs to exactly one rule family [Neuböck et al., 2013, p. 40]. The comparative cube defines the comparative facts that are in the scope of the rule. The conditions define thresholds for positive and negative activation respectively. The two different conditions are needed in order to implement the desired behaviour of hierarchical analysis and will be explained later in this section with regard to the different evaluation strategies.

An analysis rule family consists of a set of analysis rules organised in a rule hierarchy defining a single root rule and a set of scores, which contains all scores used by one of the comparative cubes of its analysis rules. An analysis rule family optionally defines a set of recommended actions that should be executed based on

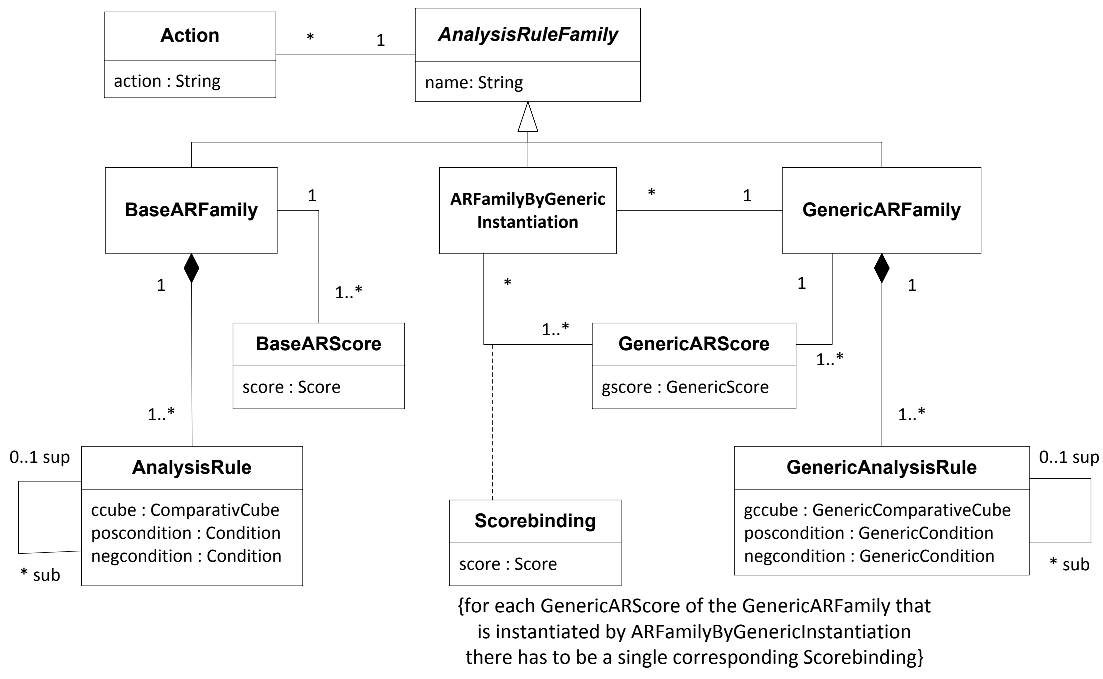
Action
action : String
AnalysisRuleFamily
name: String
BaseARFamily
ARFamilyByGeneric Instantiation
GenericARFamily
BaseARScore
score : Score
GenericARScore
gscore : GenericScore
AnalysisRule
ccube : ComparativCube
poscondition : Condition
negcondition : Condition
Scorebinding
score : Score
GenericAnalysisRule
gccube : GenericComparativeCube
poscondition : GenericCondition
negcondition : GenericCondition
0..1 sup
* sub
{for each GenericARScore of the GenericARFamily that is instantiated by ARFamilyByGenericInstantiation there has to be a single corresponding Scorebinding}

Figure 3.4: Conceptual model of analysis rules and analysis rule families

the analysis result. If an analysis rule family does not define a specific action, then the analysis result is reported to the business analyst without recommending any action, thus leaving the decision to induce a specific action to the business analyst. Analysis rule families that do not define any action are also referred to as reporting rules. The definition of actions in the context of rule families, in contrast to the definition of judgments for single judgement rules, is due to the inherent difference in the application context of the two rule types. Rather than judgements, which concern individual facts, actions and reports are compiled for a specific set of facts at a time whereby the evaluation result for a specific fact depends on the evaluation of its roll-up fact.

Figure 3.4 shows the structure of analysis rule families. For more information on generic analysis rule families and instantiations thereof see section 3.3.

In order to evaluate analysis rule families the user has to specify a number of parameters: (1) a comparative cube specifying the set of comparative facts that should be evaluated; (2) a set of analysis rule families to evaluate; (3) a list of granularities, from coarse to fine, which specifies the hierarchical order of the rule evaluation; and (4) one of two evaluation strategies, presumed or prerogative. The result of the evaluation depends on these parameters and contains the comparative points for which an analysis rule fired according to the specified evaluation strategy.

Analysis rule families can be described in terms of the ECA model as follows:

The event of an analysis rule family is either the activation event of a defined trigger or the explicit evaluation call that contains the defined parameters needed for evaluation. The condition part is the combination of the activation conditions of the rules belonging to the rule family, the provided evaluation strategy, and the provided set of granularities, which defines the evaluation path. Finally, the action that the condition of a judgement rule triggers, consists of the actions specified by the rule family or a report if no specific action is defined.

Analysis rules are specified similar to judgement rules, but in contrast to judgement rules they do not define a judgement and define two conditions instead of just one. The same organisation of rules into rule families applies to analysis rules. The two distinct conditions are used by the different evaluation strategies and are required in order to reduce information overload in the context of hierarchical data analysis. In general, if the positive activation condition is satisfied, then the action should be executed or the point should be reported. On the other hand, negative activation outlaws action execution or reporting.

Evaluation of analysis rule families is consistent with judgement rule evaluation in the sense that for each comparative point the most specific rule of a rule family is evaluated. Additionally, analysis rule evaluation requires more sophisticated rule evaluation strategies in order to apply multi-dimensional data analysis. In contrast to the implicit evaluation of judgement rules, analysis rule evaluation has to be explicitly triggered by a method call or a defined trigger. The evaluation call has to contain the previously listed inputs, which are necessary for rule evaluation.

Due to the inclusion of roll-up relations in the analysis process it is necessary to impose further restrictions to analysis rule hierarchies. In addition to the definition of a single root rule, an analysis rule family must not contain rules with overlapping rule scopes in different sub-hierarchies. Such overlaps would lead to an undefined evaluation behaviour, as analysis rule evaluation for a single fact depends on the evaluation result of its roll-up fact along the analysis path. Depending on the concrete example, overlaps might lead to two different evaluation results for the same fact. Multiple diverging results for a single fact would prevent a defined evaluation for facts that roll up to this fact.

Example 20 (Analysis Rule). Analysis rule family arCostRatio (see figure 3.1) consists of two analysis rules arCostRatio2012 and arCostRatio2012UA. The rule family does not define a specific action, and therefore is considered a reporting rule. The rule is defined to report facts with extraordinarily high increases in drug prescription costs from year 2011 to 2012, and therefore high values of RatioOfDrugPrescCosts, for insurants aggregated to the dimension role levels district and province. For the province of Upper Austria and districts thereof the more specific rule arCostRatio2012UA is defined. The result of an evaluation of this rule family depends on the provided inputs for the evaluation function. During evaluation for comparative points that are contained in the cube CostRatio2012UA, the conditions defined by analysis rule arCostRatio2012UA apply. The conditions defined by rule arCostRatio2012 apply to comparative points that are contained in the comparative cube CostRatio2012 and do not appear in cube CostRatio2012UA.

For evaluation of analysis rules a concept commonly applied by government legislation and administration bodies is used. Schrefl et al. [2013] showed the applicability of this decision scope approach to specialisation of business rules and its application in data warehousing. This approach is used as evaluation strategy for the hierarchical analysis of comparative facts.

The main idea of the decision scope approach as defined in Schrefl et al. [2013] is that different decisions can be made at different hierarchy levels. Much like in lawmaking, laws on higher administration levels have precedence over laws on any lower level, and therefore outlaw contradicting laws. A decision scope, however, does not regulate the correct decision for every fact but merely defines a frame for the next lower administration level to operate in. Decisions on lower hierarchy levels must operate within the decision scope granted by the higher hierarchy level.

Application of the decision scope approach leads to analysis rules defining two independent hierarchies at the time of rule evaluation. The rule hierarchy within a rule family expresses the knowledge that different conditions apply to different data. Rules within this hierarchy are not limited to a single granularity level and can therefore express multi-dimensional differences. In contrast, the granularity list needed for analysis rule evaluation does not impose any assertions on rule conditions but merely defines the specific evaluation path. Nevertheless, this evaluation path can have important effects on the result of the evaluation as it defines the hierarchy of precedence for analysis rule evaluation.

The basic idea of the decision scope approach is used in order to decide during the evaluation of an analysis rule whether the action, for example, to report the fact, should be executed or not. As described above, each analysis rule has two activation conditions, one negative and one positive, which are evaluated in order to make this decision. From the decision scope perspective, if a fact satisfies the rule's positive activation condition, it implies execution of the rule's action for all facts on lower hierarchy levels that roll up to this fact. In a sense the same is true for a fact satisfying the negative activation condition, only that this leads to the opposite effect; the rule's action is not executed. If neither activation condition is true for a given fact, then the activation conditions for facts on the next lower hierarchy level have to be evaluated, therefore modelling the desired behaviour that undecidable decisions are relegated to the next finer decision level.

Schrefl et al. [2013] propose two different evaluation strategies that employ the presented decision scope approach, presumed and prerogative evaluation. Both approaches are implemented by our prototype for use in analysis rule evaluation.

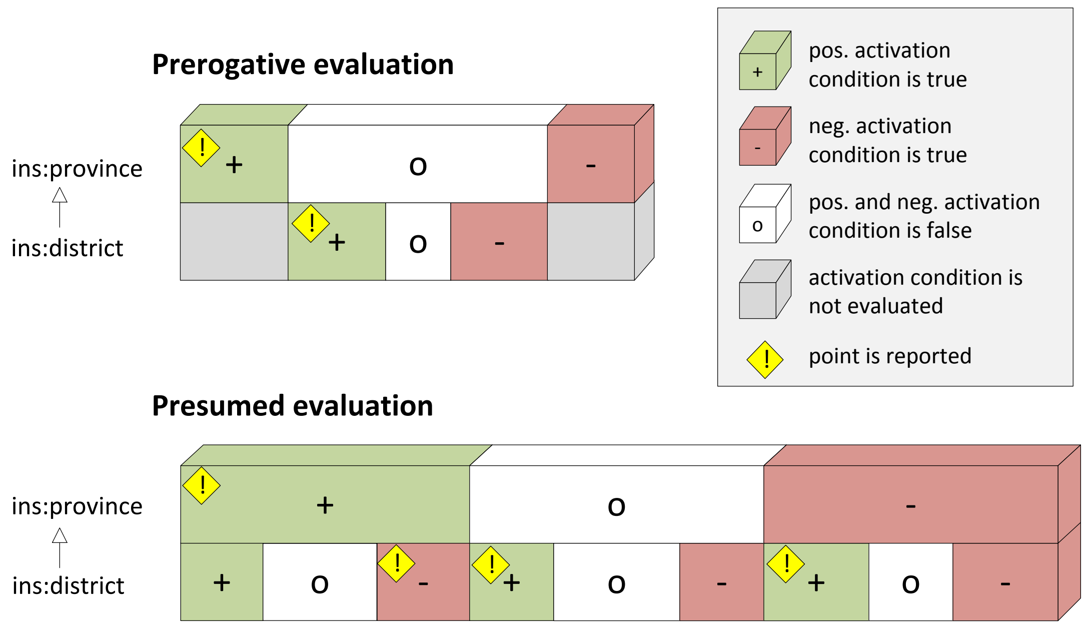
Prerogative evaluation
ins:province
ins:district
+
o
-
Presumed evaluation
pos. activation condition is true
neg. activation condition is true
pos. and neg. activation condition is false
activation condition is not evaluated
point is reported

Figure 3.5: Analysis rule evaluation strategies

3.2.1 Prerogative Evaluation

The prerogative evaluation strategy [Schrefl et al., 2013] states that rules defined on a higher or more general level always precede rules on lower levels, and therefore constitutes a top-down evaluation. This is true for positive activation as well as for negative activation. Therefore, if either the positive or the negative activation condition is satisfied by a fact on the top granularity level, evaluation for this fact as well as for facts that roll up to this fact stops. Rules on the next lower granularity level as defined by the evaluation path, are only evaluated for facts that roll up to a previously undecided fact, that is, neither positive nor negative activation condition evaluated to true.

Starting at the first granularity the set of facts on that granularity is evaluated against the analysis rule family. This results in a division of facts into three sets. The first set contains all facts that satisfy the positive activation condition. These are the facts for which the defined action is triggered. The second set contains all facts that satisfy the negative condition; no action is executed for these facts. Evaluation does not continue for facts that roll up to one of the facts contained in either of these two sets, as the same activation is assumed for subsequent granularities. The third set is the set of facts that neither satisfy the positive nor the negative activation condition; no action is executed on this granularity level, but a further detailed analysis is triggered for all facts that roll up to one of these undecided facts. On the next finer granularity level the same principle is applied to the remaining facts that roll up to a previously undecided fact. If no activation condition is satisfied for a fact and no more granularities are defined in the analysis path, then no action is recommended for this fact.

Example 21 (Prerogative Evaluation). The top part of figure 3.5 shows the basic idea of the prerogative evaluation approach. Suppose the previously defined analysis rule arCostRatio (see figure 3.1) is evaluated using the comparative cube CostRatio2012, the strategy prerogative and the list of granularities {[ins:province], [ins:district]}. Starting at the granularity [ins:province] the set of facts on that granularity is evaluated against the analysis rule family arCostRatio. Suppose a comparative fact for the province of Lower Austria with a score value of 1.3, which results in a positive activation. Therefore, the point of Lower Austria is reported and no detail analysis is conducted for the distinct districts of Lower Austria as the same positive activation is implied for Lower Austrian districts. In contrast, a comparative fact for the province of Vienna with a corresponding score value of 0.9 leads to negative activation. Therefore, the point is not reported and as in the case of Lower Austria, no detail analysis is conducted. Finally, consider the province of Upper Austria with a score value of 1.3 (note that the more specific rule arCostRatio2012UA applies to Upper Austria and Upper Austrian districts). This value evaluates to neither a positive nor a negative activation. Therefore, no action for the province of Upper Austria is reported, however, a detail analysis can be conducted on the next granularity level along the specified evaluation path, [ins:district]. Therefore, for districts in Upper Austria the evaluation continues and the districts that satisfy the positive activation condition are reported. For example, the Upper Austrian district Linz-Stadt with a corresponding score value of 1.5 would trigger the defined action. A different district Wels-Stadt with a score value of 1.3 does satisfy neither activation condition and is not reported. In general, this would lead to a more detailed analysis on the next finer granularity level. However, the defined analysis path does not contain further granularities. Therefore, no further detail analysis is possible.

3.2.2 Presumed Evaluation

An alternative evaluation strategy is the presumed evaluation. Schrefl et al. [2013] argue, that this strategy is best suited for reports as it allows to augment the results of prerogative evaluation with detailed results on finer granularities. The underlying idea is that only those results on finer granularities that contradict the previously reported evaluation on a coarser granularity are reported. This behaviour enables detailed insights for business analysts, while preventing information overload by only reporting contradicting results.

Starting at the first granularity, the set of facts on this granularity is evaluated against the analysis rule family. Contrary to the presumed evaluation strategy, rule evaluation does not stop for facts that satisfy an activation condition. By using the presumed evaluation strategy, evaluation continues on each additional granularity defined by the analysis path. However, on finer granularities an action or report is only triggered for those facts that contradict a previous activation of a roll-up fact by satisfying the opposite activation condition.

Example 22 (Presumed Evaluation). Suppose the evaluation of analysis rule family arCostRatio (see figure 3.1) using the presumed evaluation strategy on the comparative cube CostRatio2012 with the analysis path {[ins:province], [ins:district]}. The comparative fact for the province of Lower Austria with a score value of 1.3 results in a positive activation. Therefore, the province of Lower Austria is reported as in the case of prerogative evaluation. Though, in contrast to the prerogative strategy, analysis does not stop but is also conducted for districts of Lower Austria in order to check for contradicting activations on the district level. Therefore, on granularity level [ins:district] all Lower Austrian districts are evaluated in order to find districts that satisfy the negative activation condition and would therefore contradict the positive activation of their roll-up fact. If such a fact satisfies the negative condition, then the district is reported in order to indicate a contradiction to the activation for Lower Austria on level [ins:province]. Conversely, if a province, for example, Vienna, satisfies the negative activation condition, then Vienna districts are evaluated in order to find and report the districts with contradicting positive activation.

A variation of the presumed evaluation strategy is to define a granularity limit for the report of contradicting facts. It is conceivable that a contradicting activation is more significant for facts that lie directly below the fact responsible for the initial activation compared to facts that lie two or three granularity levels below the initial fact in the roll-up hierarchy. Therefore, it might be desirable to only report contradicting activations if they lie directly below the initial activation. Contradictions on even lower granularity levels are not relevant for specific use cases and do not provide valuable information to the business analyst and might be omitted in order to further reduce information overload.

For some specific use cases it might be appropriate to implement additional evaluation strategies that employ a hybrid approach by combining the discussed prerogative and presumed evaluation. For example, a presumed positive evaluation strategy can be defined that uses the prerogative evaluation strategy for facts matching the negative activation condition, that is, no further evaluation for facts that roll up to a fact with negative activation; for facts matching the positive activation, however, the presumed evaluation strategy is employed by evaluating facts that roll up to facts satisfying positive activation in order to find contradicting facts with negative activation.

In the presented prototype implementation the two discussed evaluation strategies have been implemented in their pure form as described in this section.

3.3 Generic Rules

In this section we extend the previously defined concepts for rules by introducing generic judgement and analysis rules as well as the corresponding generic rule families. First, we discuss the motivation behind our concept of generic rules. Then, the definition of generic rules based on generic scores and generic comparative cubes is defined. We also provide information on how to instantiate generic rule families for specific concrete analysis situations. Further, we show the differences between generic and non-generic rules and discuss the resulting consequences for the evaluation of a generic rule family.

Generic rules, much like generic measures and generic scores, avoid the necessity to define similar concepts multiple times from scratch by providing a concept that can be defined using variables or generic qualifiers instead of concrete concepts. Additionally, the concept of generic rules allows to define rules that apply to generic analysis situations as previously defined in section 2.4. Therefore, instead of limiting the scope to a specific analysis situation, generic rules allow to define a broader, generic scope.

Two generic rule types, generic judgement rule and generic analysis rule, are distinguished. Each of these types defines the generic embodiment of their non-generic counterparts. Generic rules are organised in corresponding generic rule families, that is, generic judgement rule family and generic analysis rule family. Each generic rule family consists of a set of generic rules that is organised in a rule hierarchy based on the hierarchy of the rule's scopes. As the scopes of generic rules are defined by generic comparative cubes, the generic rule hierarchy is based on generic comparative cubes, too. A generic rule family defines a set of generic scores, contrary to base rule families, which define only non-generic scores.

The condition of generic rules, analogous to non-generic rules, is defined using generic score-value-comparisons defined over generic scores contained in its generic comparative cube. Similar to non-generic conditions, multiple comparisons of generic scores can be concatenated by boolean expressions (see figure 3.2).

Generic and non-generic rules differentiate in their definition as each generic rule (1) belongs to a generic rule family, (2) defines a generic comparative cube and (3) defines its condition in terms of generic score-value-comparisons. Generic rules are evaluated against non-generic analysis situations as generic analysis situations cannot be executed in the semDWH due to their generic nature.

Generic rule families, like all other generic concepts, have to be instantiated in order to allow a translation into the semDWH. This implies that only instantiations of a generic rule family can be evaluated by the rule engine. Therefore, in order to compute an evaluation result for a generic rule family, the generic rule family has to be instantiated beforehand. The scores defined by the concrete analysis situation are used to match each generic score defined by the generic rule family with an instantiation of the score defined by the analysis situation. With the concrete scores of the analysis situation the generic rule family can be instantiated. Further, the instantiated rule family can then be evaluated against the analysis situation.

Together with the instantiated scores, the generic comparative cubes of the rules in the rule family can be instantiated as long as the actual qualifiers used comply with the generic qualifier domains specified by the generic comparative cubes. If the actual qualifiers do not comply with the specified cube domain, then the generic cube is not instantiated. As a consequence, the rule defined based on this specific generic comparative cube does not apply to any facts of the analysis situation and is therefore ignored for this rule family instantiation. The generic scores used for the definition of the rule conditions are substituted with the concrete scores for rule evaluation and translated into concrete score-value-comparisons. After the instantiation of all generic components the now instantiated rule family can be translated into the semDWH and used for evaluation. For each subsequent use of the generic rule family this instantiation process is executed again with the scores defined by the respective analysis situation. Note, that there is no explicit instantiation for generic rules, but only for generic rule families. Instantiations of generic rules are implicitly defined by the definition of the rule family that instantiates a generic rule's rule family.

Example 23 (Generic Rule Family Instantiation). Figure 3.6 depicts the conceptual result of the instantiation of generic judgement rule family gjrAvgCostIncr defined over the generic score gRatioOfDrugPrescCosts with the concrete score RatioOfDrugCostsInDrug, which in turn is defined as instantiation of the generic score gRatioOfDrugPrescCosts with the concrete qualifier InDrug. The instantiated rule family is defined as non-generic judgement rule family by generic instantiation of judgement rule family gjrAvgCostIncr and the binding of the generic score gRatoOfDrugPrescCosts to the non-generic score RatioOfDrugPrescCostsInDrug. The generic rule family comprises two rules defined on generic comparative cubes with different domains for their generic qualifiers. Generic comparative cube gCostRatio2012 defines the generic qualifier domain using a md-metaconcept defined by the subsuming concept InDrug compared to a md-metaconcept defined by the subsuming concept InOAD for the generic qualifier domain of cube gCostRatio2012UA. Due to the defined generic qualifier domains, only rule gjrAvgCostIncr2012 is applicable for the score RatioOfDrugPrescCostsInDrug. The scope of rule gjrAvgCostIncr2012UA defined by the generic cube gCostRatio2012UA does not comply with the concrete score and its qualifier, as the concept InDrug is not subsumed by concept InOAD and is therefore not in the scope of the cube. The interpretation of the instantiation result is a non-generic judgement rule family consisting of a non-generic rule representing the instantiation of the rule gjrAvgCostIncr2012 with the score RatioOfDrugCostsInDrug.

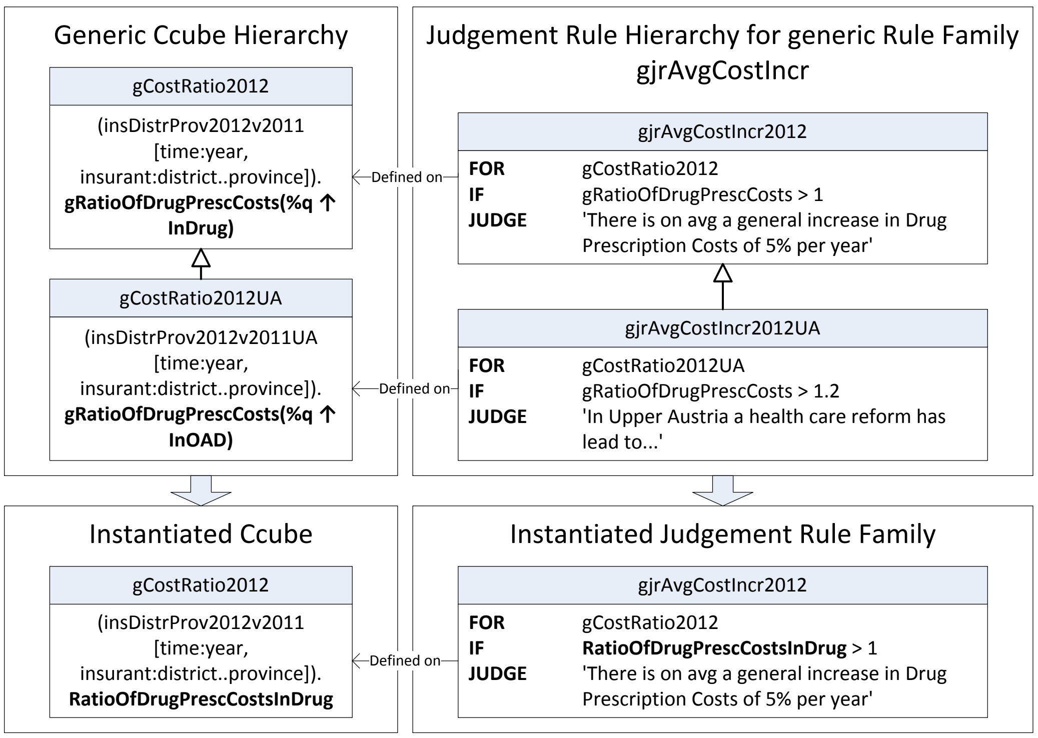
Generic Ccube Hierarchy
gCostRatio2012
(insDistrProv2012v2011 [time:year, insurant:district..province]). gRatioOfDrugPrescCosts(%q ↑ InDrug)
gCostRatio2012UA
(insDistrProv2012v2011UA [time:year, insurant:district..province]). gRatioOfDrugPrescCosts(%q ↑ InOAD)
Defined on
Judgement Rule Hierarchy for generic Rule Family gjrAvgCostIncr
gjrAvgCostIncr2012
FOR gCostRatio2012
IF gRatioOfDrugPrescCosts > 1
JUDGE 'There is on avg a general increase in Drug Prescription Costs of 5% per year'
gjrAvgCostIncr2012UA
FOR gCostRatio2012UA
IF gRatioOfDrugPrescCosts > 1.2
JUDGE 'In Upper Austria a health care reform has lead to...'
Instantiated Ccube
gCostRatio2012
(insDistrProv2012v2011 [time:year, insurant:district..province]). RatioOfDrugPrescCostsInDrug
Instantiated Judgement Rule Family
gjrAvgCostIncr2012
FOR gCostRatio2012
IF RatioOfDrugPrescCostsInDrug > 1
JUDGE 'There is on avg a general increase in Drug Prescription Costs of 5% per year'

Figure 3.6: Conceptual instantiation of generic judgement rule family gjrAvg-CostIncr with score RatioOfDrugPrescCostsInDrug

The instantiation of a generic rule family yields a non-generic rule family by generic instantiation. Evaluation of the instantiated rule family is analogous to the evaluation described in the previous sections for judgement rules and analysis rules. Even though the evaluation is equivalent once the generic rule family has been instantiated, there are differences in the evaluation process between generic and non-generic rule families prior to the instantiation. These differences will be discussed in the following subsections together with an explicit definition of the generic rule types.

3.3.1 Generic Judgement Rules

A generic judgement rule is defined by (1) a generic comparative cube, (2) a generic condition (3) a corresponding textual judgement, and (4) belongs to exactly one generic judgement rule family. The judgement, just like for non-generic judgement rules provides the information or annotation that is reported for facts that satisfy the rule condition. The judgement text is thereby independent of the concrete analysis situation for which the rule is evaluated.

Due to the fact that generic judgement rules are defined using generic comparative cubes and generic scores, the process for deciding whether a generic judgement rule family is evaluated changes. Similar to the decision for non-generic judgement rules both the scores and the comparative concept have to be examined for possible overlaps. For each generic score of the rule family definition a non-generic score that is defined as instantiation of that generic score has to be contained in the analysis situation in order to allow for a match. If one or more of the generic scores do not have a non-generic counterpart in the analysis situation, the generic rule does not apply to the analysis situation and is therefore neither instantiated nor evaluated. The comparative concepts of the generic comparative cube of the root rule and the comparative cube of the analysis situation are used in order to decide whether an overlap of comparative points can be eliminated. This process does not change for generic judgement rules. Combining the results of these comparisons a set of generic judgement rule families remains for which an overlap cannot be eliminated. These remaining rule families are therefore instantiated and their instantiations are subsequently evaluated for the analysis situation analogous to the other applicable judgement rule families.

3.3.2 Generic Analysis Rules

A generic analysis rule is defined accordingly by (1) a generic comparative cube; (2) two generic conditions, one for positive and one for negative activation; and (3) belongs to exactly one generic analysis rule family.

Evaluation of generic analysis rule families is explicitly called as in the case of non-generic analysis rule families. The parameters for this explicit evaluation call do not change. The list of analysis rule families that should be evaluated can contain both generic and non-generic analysis rule families. During rule evaluation, all generic analysis rule families that are applicable to the concrete analysis situation are instantiated. If a generic analysis rule family cannot be instantiated based on the analysis situation, it is removed from further rule evaluation. A generic rule family cannot be instantiated if a generic score of the generic rule family does not have a matching non-generic score in the analysis situation, or if the qualifier domain of the rule family's root rule is violated by a qualifier of one of the non-generic scores of the analysis situation. If additional analysis rule families that do apply to the analysis situation have been provided for evaluation, they are evaluated normally. After instantiating the generic analysis rule families, the instantiated rule families together with the defined base analysis rule families are evaluated against the analysis situation as described in section 3.2.

3.4 Discussion

In this chapter we described generic and non-generic judgement and analysis rules and how they can be defined based on MDO concepts.

Judgement rules are used to annotate judgements to comparative data analysis situations and are similar to threshold alerts provided by state-of-the-art data warehousing systems. Judgement rules, however, allow to define more sophisticated rules in the context of comparative data analysis. Judgements defined by judgement rules are automatically annotated to the results of comparative analysis situations. Structuring of judgement rules in rule hierarchies enables overriding of general rules with more specific ones. In addition, generic judgement rules can be used to define judgements for generic analysis situations.

Analysis rules perform hierarchical analysis of multi-dimensional comparative data and can be evaluated using different evaluation strategies based on the decision scope approach [Schrefl et al., 2013] in order to prevent information overload and obtain relevant information depending on the current use case. Analysis rules can define specific actions that should be executed based on the evaluation result. At this stage the actions defined by analysis rules merely provide hints on recommended actions whereas their execution together with associated tasks like conflict resolution is left to the business analyst. Further remarks on this topic are provided in section 4.7.

4 Rule Engine Implementation

In this chapter we describe the implementation of the rule engine functionality as part of the semCockpit system prototype. The approach is implemented using the Oracle DB 11.2g database technology. The prototype architecture is geared towards functionality, rapid prototyping and experimentation. Performance considerations are widely neglected.

This chapter starts with a brief description of the implementation of the core semCockpit components and the semCockpit system architecture. After the core components we concentrate on the implementation of the rule engine. We show how rules and rule families are represented within the MDO-DB, how they are implemented in the semDWH and how the translations between these two components are derived. Then, we describe the implementation of the evaluation algorithms for judgement and analysis rules. After that, the results of a preliminary performance study for the rule engine implementation are presented. Finally, we discuss the presented prototype implementation and its limitations.

4.1 semCockpit Architecture

In this section we provide an overview of the architecture of the semCockpit system prototype and the technologies used for its implementation.

Figure 4.1 depicts the semCockpit system architecture and shows the main physical components and technologies used. The main components, MDO-DB and semDWH are each implemented in a separate Oracle Database schema. Most of the additional components have been implemented as PL/SQL packages located within the MDO-DB schema. These include the rule engine and the MDO-DWH Mapper. The semCockpit reasoner is partly implemented using PL/SQL as well but also consists of a small Java program in order to be able to use an off-the-shelf OWL reasoner for Java. The frontend is implemented as a web-based interface.

The semDWH component of the prototype has been created from scratch by defining the necessary SQL DDL statements. The data warehouse has been filled with a factitious data set. Consistent initial states for MDO-DB and semDWH are obtained through an initialisation process, which creates the necessary MDO Base definitions, for example, dimensions, levels, and base measures.

The base workflow of the system prototype is as follows: (1) The user defines new

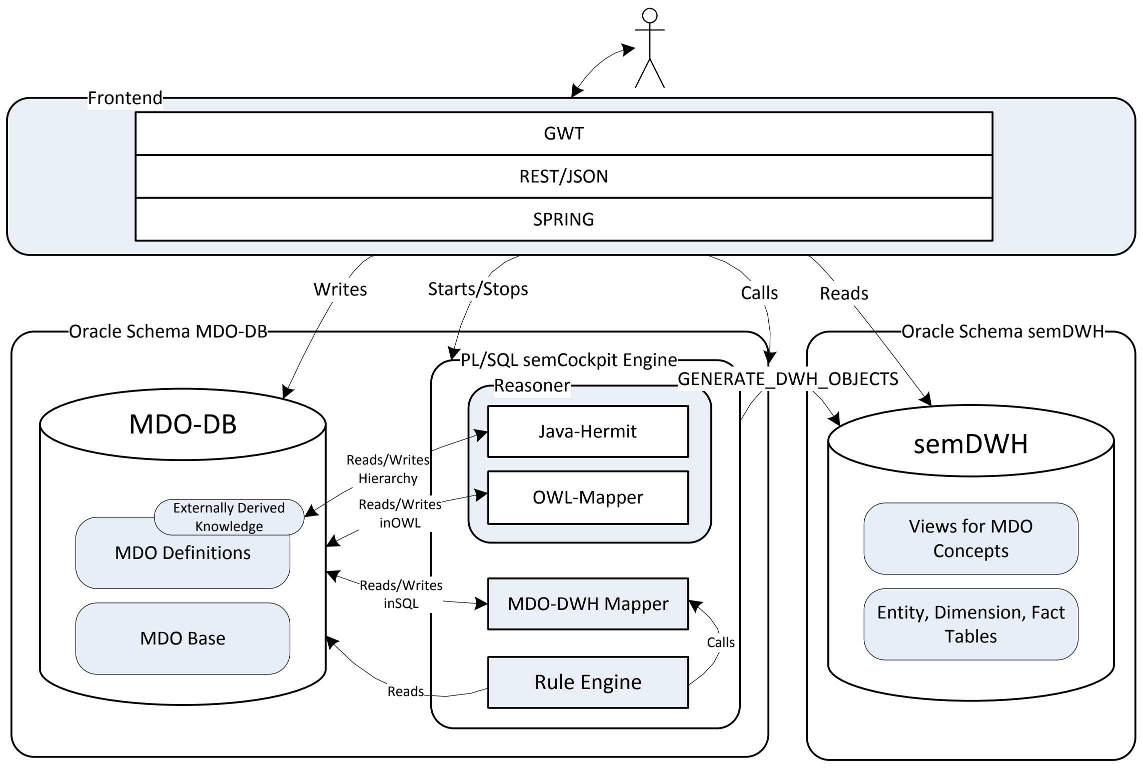

Figure 4.1: semCockpit system architecture

concepts or builds a new analysis query using the user interface. (2) After completing the definition the user saves the created concepts, which leads to the frontend writing the concept definitions into the MDO-DB. (3) Triggers on the MDO-DB ensure that corresponding SQL statements for the realisation of the defined concepts in the semDWH are created and persisted by the MDO-DWH Mapper. In the semDWH, MDO concepts are, depending on the concept type, realised as views or materialised views over the underlying data warehouse. (4) The frontend triggers the process for executing the SQL statements for the newly defined MDO concepts in the semDWH. (5) The frontend reads the defined concepts or analysis queries from the semDWH by querying the concept views and displays the result to the user.

In order to add judgement and analysis rules to the semCockpit system prototype, extensions of the MDO-DB and the MDO-DWH Mapper were necessary. The MDO-DB stores the definitions for all explicitly defined concepts, and therefore stores the definitions of rules and rule families as well. Thus, a model for rule definitions had to be added to the MDO-DB. The MDO-DWH Mapper, as component that manages the translation and communication between MDO-DB and semDWH, has to handle the same functionality for rules as well. Therefore, rule specific extensions to the MDO-DWH Mapper were necessary. In the rule engine module the evaluation process for rules is encapsulated. It uses the rule specific extensions to the other components in order to compute rule evaluations and provides an interface for the invocation of rule evaluations to the semCockpit frontend. The structure of the semDWH has not been affected by the rule engine implementation.

4.2 Rule Representation in the MDO-DB

Rules and all other object definitions needed in the context of the rule engine are represented as tuples in relational tables in the MDO-DB. Rule types are represented by different relations sharing some similarities. These relational schemata will be presented in this section. Within the MDO, rules and rule families are fully defined through their attributes and references to other MDO constructs as defined in chapter 3.

condition(conditionID = (conditionName), *rootCondtermID*)
conditionTerm(condtermID = (*conditionID*, sequenceNr), discriminator)
simpleCondition(*condtermID*, *scoreID*, compOperator, compValue)
complexCondition(*condtermID*, *leftCondtermID*, *rightCondtermID*, concatOperator)
simpleGenericCondition(*condtermID*, *gscoreID*, compOperator, compValue)

Table 4.1: MDO-DB representation of rule conditions

The different types of rules have one thing in common: the rule condition. Rule conditions are composed of one or more score-value-comparisons, which are concatenated by simple boolean expressions. Depending on whether the condition is for a generic or a non-generic rule, the referenced score is a generic or non-generic score. This leads to the following relational representation of a rule condition (see table 4.1).

Relations are represented by a set of attributes enclosed by brackets. A derived attribute defines through a bracketed equation term the attributes from which it is derived. The attributes that form the primary key of a relation are underlined; foreign keys are set in italics.

The base relation for a rule condition defines an identifier for the condition and a root condition term. There are three different types of condition terms. (1) simple condition terms, (2) complex condition terms, and (3) simple generic condition terms. Types 1 and 3 only differ in the referenced score type, that is, non-generic for simple conditions and generic for simple generic conditions, and are defined by a single score, a comparison operator, and a single comparison value. Complex condition terms represent the concatenation of two simple terms and define a left condition term, a concatenation operator, and a right condition term.

Rule families are defined by a rule name and a type discriminator defining whether a rule family consists of generic rules, non-generic rules, or is defined as instantiation of a generic rule family. Additionally, in a separate relation, for each rule family the scores within its scope are defined. The definition of available scores for rule families allows for easier checks for computing possible overlaps with the resulting cube of an analysis situation. Judgement and analysis rule families are maintained in separate relations. Score references for generic and base rule families are stored in separate relations, too, as generic rule families define generic scores whereas base rule families define non-generic scores. Rule families defined by instantiation of a generic rule have to store score bindings, which relate each generic score of the generic rule family to a non-generic score that is used for rule instantiation.

Rule hierarchy tables store the subrule-superrule relations of rules within the same rule family. The entries for these tables are derived from the reasoning component. There are four hierarchy tables, one for each of the different rule types. We assume that a hierarchy reasoning method that populates the provided rule hierarchy tables with correct reasoning results for rules is available in some form. If no reasoning functionality is implemented, the rule hierarchies have to be provided manually by the user.

judgementrulefamily(jrulefamilyID = (name), discriminator)

basejudgementrulescore(*jrulefamilyID*, *scoreID*)
basejudgementrule(jruleID = (*jrulefamilyID*, *ccubeID*), judgement, *conditionID*)
basejudgementrulehierarchy(*sub_jruleID*, *sup_jruleID*)

bygenericinstantiation(*jrulefamilyID*, *gjrulefamilyID*)
scorebinding(*jrulefamilyID*, *gjrulescoreID*, *scoreID*)

gjudgementrulescore(gjrulescoreID = (*jrulefamilyID*, *gscoreID*))
genericjudgementrule(gjruleID = (*jrulefamilyID*, *gccubeID*), judgement, *conditionID*)
gjudgementrulehierarchy(*sub_gjruleID*, *sup_gjruleID*)

Table 4.2: MDO-DB representation of judgement rules and judgement rule families

All judgement rule families that are currently contained in the MDO are stored in the judgementrulefamily relation. A discriminator denotes for each family whether it is a base judgement rule family, a generic judgement rule family or a judgement rule family by generic instantiation.

Base judgement rule families consist of rules represented by the basejudgementrule relation. Each judgement rule has to define a reference to a comparative cube defining the scope of the rule. The rule family name together with the identifier of the comparative cube form the identifier for a judgement rule. Additionally, judgement rules define a judgement in textual form, which defines the actual judgement that is shown for facts that activate the rule. Finally, a reference to a rule condition models the rules activation condition. As shown in table 4.2, the hierarchy of judgement rules is stored in table basejudgementrulehierarchy. As addressed before, the set of scores of a base judgement rule family is stored in the basejudgementrulescore relation.

The definitions for a generic judgement rule family and its containing rules are stored in a different set of relations, genericjudgementrule, gjudgementrulescore and gjudgementrulehierarchy. The genericjudgementrule relation consists of a reference to a generic rule family from the rule family relation, a reference to a generic comparative cube, a reference to a condition and, like non-generic judgement rules, a textual judgement. Therefore, the only differences between the genericjudgementrule and basejudgementrule relation are the different references to generic concepts for the comparative cube and the condition of a generic judgement rule. The relation gjudgementrulehierarchy captures the derived rule hierarchy information for generic judgement rules. Finally, in relation gjudgementrulescore the generic scores of a generic judgement rule family are defined.

The instantiation of a generic judgement rule family is represented as a specific type of rule family with the defined discriminator byGenericInstantiation and is defined based on the generic rule family that it instantiates. This relationship is recorded in the relation bygenericinstantiation. In order to fully specify a generic rule instantiation, the relation scorebinding has to define a non-generic score for each of the generic scores of the generic rule family. This binding information together with the initial definition of the generic rule family sufficiently defines an instantiated rule family in the MDO-DB.

Analysis rule families, just like judgement rule families, are stored in a single relation analysisyrulefamily. This relation contains the currently defined analysis rule families together with the respective family type. These are base analysis rule family, generic analysis rule family and analysis rule family by generic instantiation. Analysis rule families can also define an action, which, depending on the evaluation strategy, should be executed for facts that fulfil the positive condition of an analysis rule within the rule family. Actions are kept in separate relations and each action tupel references the analysis rule family to which it applies (see table 4.3). If no specific action is defined, the rule engine assumes a reporting action, which returns the facts as evaluation results without suggesting a specific action.

The baseanalysisrule relation contains the necessary definition of a non-generic analysis rule. As for judgement rules each rule defines a comparative cube and

analysisrulefamily(arulefamilyID = (name), discriminator)
action(*arulefamilyID*, action)

baseanalysisrulescore(*arulefamilyID*, *scoreID*)
baseanalysisrule(aruleID = (*arulefamilyID*, *ccubeID*), *posconditionID*, *negconditionID*)
baseanalysisrulehierarchy(*sub_aruleID*, *sup_aruleID*)

bygenericinstantiation(*arulefamilyID*, *garulefamilyID*)
scorebinding(*arulefamilyID*, *garulescoreID*, *scoreID*)

ganalysisrulescore(garulescoreID = (*arulefamilyID*, *gscoreID*))
genericanalysisrule(garuleID = (*arulefamilyID*, *gccubeID*), *posconditionID*, *negconditionID*)
ganalysisrulehierarchy(*sub_garuleID*, *sup_garuleID*)

Table 4.3: MDO-DB representation of analysis rules and analysis rule families

belongs to a rule family. Analysis rules store references to two conditions, one triggering positive activation, and one for negative activation. Analogous to non-generic judgement rules, the tables baseanalysisrulehierarchy and baseanalysisrulescore store the hierarchy of rules within a rule family, and the scores that appear in a rule family, respectively.

Accordingly, generic analysis rule definitions are stored in their own relation genericanalysisrule, which defines the rule's generic analysis rule family, its scope in form of a generic comparative cube and its two conditions. In the relation ganalysisrulehierarchy information about the hierarchical ordering of rules within the same generic rule family are stored and the relation ganalysisrulescore holds the generic scores for which the generic analysis rule family is defined.

The instantiation of a generic analysis rule family is represented analogous to the instantiation of a judgement rule family by defining an additional rule family type and storing the binding information in separate relations. The rule family is defined as analysis rule family with the discriminator byGenericInstantiation and defines generic rule family that is instantiated in the relation bygenericinstantiation. Additionally, the relation scorebinding holds the bindings of non-generic scores to the generic scores of the rule that is instantiated in the same way as for the instantiation of a generic judgement rule family.

4.3 Rules in the semDWH

Rules, like other MDO concepts, are represented as views over the underlying data of the semDWH. Neumayr et al. [2013] describe the transformation of MDO concept definitions to semDWH views.

A rule view consists of the facts of the comparative cube used for rule definition that fulfil the rule's condition. Depending on the rule type some additional information, for example, the judgement of a judgement rule, is part of the rule view. Rule views represent the interpretation of single rules without considering their hierarchical order within a rule family. Rule hierarchies of rule families are represented in rule family views also referred to as aggregated rule views.

A judgement rule view consists of those facts of its comparative cube that fulfil the rule condition. For each fact the defined textual judgement is annotated by creating a column Judgement in the rule view. To be able to track the specific rule that caused a judgement, additionally to the judgement the identifier of the rule is added to the view in a separate column Rule. Assuming that a rule does not have any other rules within its rule family a rule view can be joined to the result of an analysis situation in order to gain all facts of the analysis situation for which the rule fires together with the defined judgement and the rule identifier.

Each analysis rule is represented by two rule views in the semDWH, with one view consisting of those facts that fulfil the positive activation condition and the other view consisting of the facts fulfilling the negative activation condition. As analysis rules do not define other rule specific information the only added column to the rule view is the Rule column in order to be able to track the corresponding rule. Analysis rule actions are defined based on rule families and not for single rules. Therefore, this information is also added during evaluation and is not represented in individual rule views.

As the evaluation of rules is always based on rule families, a view is needed that aggregates all rule views of a rule family with respect to their hierarchy within the rule family. The previous paragraphs only described the information contained in a rule view for single rules independent of their hierarchical order within a rule family. As defined in section 3, for each fact only the most specific rule within a rule family should be evaluated. The rule hierarchy provides the necessary information on which rule is more specific than others in order to create an aggregated view that is consistent with this definition. An aggregated rule view is the representation of a MDO rule family in the semDWH. Note, that for analysis rule families two such aggregated rule views, one for positive and one for negative activation, are defined.

An aggregated rule view is the union of all rule views of a rule family, whereby each rule view is restricted to those comparative points for which it is the most specific defined rule within the rule hierarchy. The resulting interpretation of an aggregated rule view is a set of comparative points, annotated with the previously defined information for triggering rule and judgement for judgement rules, whereby each comparative point contained in the set satisfies the condition of the most specific rule defined for that point. If a comparative point does not satisfy this condition it is not contained in the aggregated rule view. A comparative point might be contained multiple times in an aggregated rule view for different rules, if the scope of rules in different hierarchies of the same rule family overlap. Note, that such overlaps are only allowed for judgement rule families. Overlaps in the hierarchy of an analysis rule family can lead to ambiguous or undefined evaluation outcomes.

Due to their generic nature, generic rules do not have a corresponding representation in the semDWH. Generic rules, much like generic measures and generic scores, have to be instantiated, by binding non-generic concepts to all generic elements, before a semDWH representation can be computed. The instantiation of a generic rule is achieved by defining a corresponding rule family by generic instantiation of a generic family through binding a concrete score to each of the generic scores defined by the generic rule family. The need to define a new non-generic rule family for representing the instantiation of a generic rule family is necessary as a generic rule family can have multiple instantiations, each specifying a different set of score bindings. Note, that the instantiation of generic rules requires the instantiation of a generic rule family. Single generic rules cannot be instantiated independent of their rule family.

4.4 Rule Mapping

The MDO-DWH Mapper is the component responsible for deriving correct SQL statements from MDO definitions and providing functionality for the execution of those statements in the semDWH. The Mapper is implemented using Oracle PL/SQL and is integrated in the MDO-DB schema. The decision to build the Mapper in this way is attributed to the employed prototyping approach. The PL/SQL implementation allows, through the use of triggers, for an easy and fast communication between the MDO and the mapping component.

A number of triggers on MDO relations initiates the mapping process as soon as a new object is inserted into the MDO-DB. As concepts are normally represented in different relations within the MDO-DB, it is possible that not all relevant definitions are available in the MDO at the time the mapping process is triggered. Therefore, the first step of the mapping process is to check whether all relevant definitions for the creation of the mapping are available. If some information is missing, the mapping process returns a placeholder statement for the concept, which is represented by an empty view. Further SQL insert statements then trigger the mapping process again and, if all information is available, the Mapper computes the correct

SQL DDL representation of the concept. For each concept a drop statement, which allows to remove the concept from the semDWH, is computed additionally to its create statement. These drop statements can be used to rollback to a previous semDHW state.

The computed statements are stored in the MDO-DB, split into create and drop statements, further referred to as *mapping sequence*. The chronology of concept definitions is stored in a relation in the MDO-DB. Each entry in the relation consists of a sequence number, the concept type, the concept identifier, a reference to the corresponding create statements and a reference to the drop statements. Through backward execution of the drop statements along the sequence number a previous semDWH state can be reconstructed.

The mapping component provides a name registry relation in order to cope with differences in the naming of concepts within the MDO and the semDWH. For the naming of MDO concepts the user can use arbitrary character strings that are stored as VARCHAR2 strings in the MDO-DB and are used as the identifier or part of the identifier of the concepts. Conceptually, for each MDO concept a corresponding view is created in the semDWH. Using the MDO identifier as name of the view is not feasible as most database systems enforce some constraints on the allowed structure of identifiers. For example, in Oracle it is not possible to use strings with more than 30 characters as name for a view or relation. In order to support both individual concept names and correct identifiers the mapping component provides functionality for mapping the user defined names to valid database identifiers. Those mappings are stored and can be used to translate the name of a concept in the MDO to the name of its semDWH view and vice versa.

Execution of the created mappings in the semDWH does not occur until a specific procedure, GENERATE_DWH_OBJECTS, of the MDO-DWH Mapper is called. Whether this procedure is executed immediately after a new concept is defined, in regular intervals, or in some other way is determined outside the core semCockpit system. For example, the current frontend implementation for the prototype executes the mapping procedure before a new query is executed in order to ensure that all concepts that might be referenced by the query have been created in the semDWH. This design provides some flexibility as to how close the MDO-DB and the semDWH are connected.

For the generation of rule mappings different strategies can be applied depending on the timing of the aggregated rule view mappings. The mapping of base rules independent of their rule hierarchy is straightforward and similar to the mapping of any other MDO concept. For more details on the mapping of other MDO concepts refer to Neumayr et al. [2013]. They provide a description of how to map different MDO concepts to a SQL representation.

As previously described, the mapping process is based on triggers. A trigger on each relevant relation fires if a new tupel is inserted and calls the corresponding mapping function. The mapping function checks whether all necessary information is available and, if so, computes the SQL DDL statements for creating and dropping the corresponding rule view. Finally, the computed statements are stored in the mapping sequence relation and wait for their execution in the semDWH.

```
conditionTerm ('cond01', 1, 'simplecondition')
conditionTerm ('cond02', 1, 'simplecondition')
simpleCondition ('cond01.1', 'RatioOfDrugPrescCosts', '>', 1)
simpleCondition ('cond02.1', 'RatioOfDrugPrescCosts', '>', 1.2)
condition ('cond01', 'cond01.1')
condition ('cond02', 'cond02.1')

judgementrulefamily ('jrAvgCostIncr', 'base')
basejudgementrulescore ('jrAvgCostIncr', 'RatioOfDrugPrescCosts')
basejudgementrule ('jrAvgCostIncr', 'CostRatio2012', 'There is on avg a general in-
crease in drug prescription costs of 5% per year', 'cond01')
basejudgementrule ('jrAvgCostIncr', 'CostRatio2012UA', 'In Upper Austria a health
care reform has lead to an avg increase in drug prescription costs of 20%', 'cond02')
basejudgementrulehierarchy ('jrAvgCostIncr.CostRatio2012UA', 'jrAvgCostIncr.Cost-
Ratio2012')
```

Table 4.4: MDO-DB representation of judgement rule family jrAvgCostIncr

Example 24 (Rule Mapping). Table 4.4 shows the relational representation of judgement rule family jrAvgCostIncr within the MDO-DB. The mapping of rules is triggered as soon as the tupel is inserted into the basejudgementrule relation. The mapping results for the two rules of rule family jrAvgCostIncr are listed in the top part of table 4.5. Note, that the rule identified by the comparative cube CostRatio2012 is mapped to name jrAvgCostIncr2012 in the semDWH. The more specific rule defined on the cube CostRatio2012UA is mapped to name jrAvgCostIncr2012UA. These mappings are stored in the name registry of the MDO-DWH Mapper. The bottom part of table 4.5 shows the necessary statements in order to compute the aggregated view for rule family jrAvgCostIncr, based on the knowledge that rule jrAvgCostIncr2012UA is below rule jrAvgCostIncr2012 in the rule hierarchy.

There are two possible strategies for mapping the aggregated rule views for rule families based on individual rule views by either computing the aggregated view after

```
CREATE VIEW "jrAvgCostIncr2012" AS
  SELECT c."goi_actdoc", c."goi_drug", c."goi_ins", c."goi_leaddoc", c."goi_ttime",
   c."goc_actdoc", c."goc_drug", c."goc_ins", c."goc_leaddoc", c."goc_ttime", 'jrAvg-
   CostIncr.CostRatio2012' AS "Rule", 'There is on avg a general increase in drug prescription costs
   of 5% per year' AS "Judgement"
  FROM (
    SELECT *
    FROM "CostRatio2012"
    WHERE "RatioOfDrugPrescCosts" > 1
  ) c;

CREATE VIEW "jrAvgCostIncr2012UA" AS
  SELECT c."goi_actdoc", c."goi_drug", c."goi_ins", c."goi_leaddoc", c."goi_ttime",
   c."goc_actdoc", c."goc_drug", c."goc_ins", c."goc_leaddoc", c."goc_ttime", 'jrAvg-
   CostIncr.CostRatio2012UA' AS "Rule", 'In Upper Austria a health care reform has lead to
   an avg increase in drug prescription costs of 20%' AS "Judgement"
  FROM (
    SELECT *
    FROM "CostRatio2012UA"
    WHERE "RatioOfDrugPrescCosts" > 1.2
  ) c;

CREATE VIEW "jrAvgCostIncr2012only" AS
  SELECT *
  FROM "jrAvgCostIncr2012"
  NATURAL JOIN (
    SELECT goi_actdoc, goi_drug, goi_ins, goi_leaddoc, goi_ttime, goc_actdoc, goc_drug,
     goc_ins, goc_leaddoc, goc_ttime
    FROM "jrAvgCostIncr2012"
    MINUS
    SELECT goi_actdoc, goi_drug, goi_ins, goi_leaddoc, goi_ttime, goc_actdoc, goc_drug,
     goc_ins, goc_leaddoc, goc_ttime
    FROM "CostRatio2012UA"
  );

CREATE VIEW "jrAvgCostIncr" AS
  SELECT *
  FROM "jrAvgCostIncr2012only"
  UNION
  SELECT *
  FROM "jrAvgCostIncr2012UA";
```

Table 4.5: semDWH mappings for judgement rule family jrAvgCostIncr

a change in the rule hierarchy occurs or by computing the aggregated view just before rule evaluation. The first strategy is to create a mapping for the aggregated rule view as soon as the first rule of a rule family is defined. Subsequently, the aggregated view has to be updated whenever the hierarchy of the rule family changes, that is, each time a change in a rule hierarchy relation concerning rules of the rule family occurs. If the aggregated view is not executed between changes in the hierarchy the update of the aggregated view is superfluous. The prototype implements the second strategy of computing the aggregated view just before rule evaluation. This strategy partly prevents unnecessary mappings of aggregated views. The idea is to compute the aggregated rule view just before evaluation of a rule family based on the, at this point, available rule hierarchy. This approach ensures that the aggregated rule view is only mapped when it is certain that the view is needed. For subsequent evaluation the aggregated view is computed again in order to account for possible changes in the rule hierarchy. Unnecessary mappings with this strategy occur when there are no changes in the hierarchy of a rule family between evaluations. In such a case the result of the aggregated view mapping is exactly the same as the result of the previous mapping. Recording the changes of the rule hierarchy tables can be used to avoid unnecessary mappings. If a change has occurred to the hierarchy of the rule family in question a new mapping is computed; otherwise, if the hierarchy did not change with respect to the previous computation, the previously computed mapping is used. As the cost of mapping aggregated rule views appears to be negligible (see section 4.6), our prototype does not implement the described checks and naively computes the aggregated rule view each time before a rule family is evaluated without checking if a new aggregated mapping is actually needed.

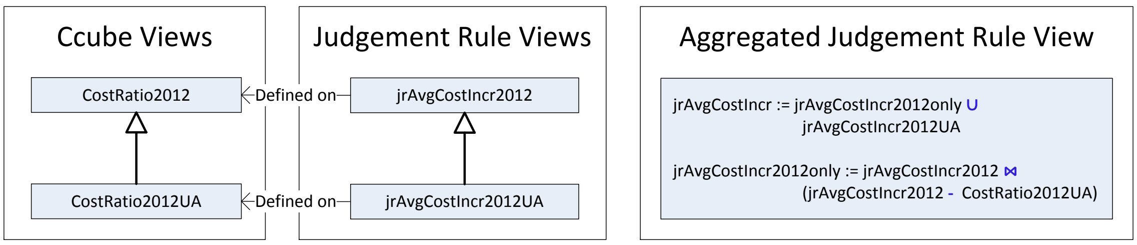
Ccube Views
CostRatio2012
CostRatio2012UA
Defined on
Defined on
Judgement Rule Views
jrAvgCostIncr2012
jrAvgCostIncr2012UA
Aggregated Judgement Rule View
jrAvgCostIncr := jrAvgCostIncr2012only ∪ jrAvgCostIncr2012UA
jrAvgCostIncr2012only := jrAvgCostIncr2012 ⋈ (jrAvgCostIncr2012 - CostRatio2012UA)

Figure 4.2: Aggregated rule view mapping

Mapping of an aggregated rule view is based on the individual rule views and the views of the comparative cubes of the individual rules. An aggregated rule view consists of the union of all individual rule views whereby all facts of a rule view that appear in the comparative cube of a rule that is below this rule in the hierarchy are removed from the aggregated view result. This results in the desired behaviour that for each fact only the most specific rule within the rule family is applied and contained in the aggregated rule view. The mapping process is currently implemented as follows (see figure 4.2): For each rule that is not a leaf in the rule family hierarchy, a subquery that contains only those facts for which the rule is the most specific rule in the hierarchy is computed. The aggregated rule view is the union of all those subqueries and all rule views of leaf rules of the hierarchy.

Generic rules require a different mapping strategy as it is not possible to generate the mappings for individual generic rules at the time of their definition. To be able to compute the mapping for a generic rule the concrete scores are needed. Therefore, the mapping for generic rules can only occur during rule evaluation, after the analysis query has been defined. The mapping for generic rules is triggered by the rule engine during rule evaluation. The analysis query, which defines the concrete scores, is used as input for the rule engine. The rule engine computes the relevant rule families and triggers their instantiation using the scores defined by the analysis situation.

Due to the possibility of qualifier domain constraints defined by the generic comparative cubes and depending on the non-generic scores used for instantiation, only a part of the hierarchy of a rule family might apply to a specific analysis situation, and therefore has to be mapped. If the concrete qualifiers of the scores are not consistent with the qualifier domain of a rule's generic comparative cube, the rule is not applicable, and therefore neither the rule nor its cube has to be instantiated and mapped. Furthermore, due to the assumption that rule and cube hierarchies are consistent and conflict free, all rules that lie below this rule in the rule hierarchy are also not applicable as well. This insight guides the implemented mapping strategy as not applicable rules do not need to be considered for the mapping of the aggregated rule family view.

Instantiation and mapping of generic rules is a two stage process. First, the rule engine evaluates the domains of the generic qualifiers of the generic cubes within an applicable rule family. Those cubes which domain is consistent with the scores employed by the analysis situation are instantiated using the scores of the analysis situation. This instantiation triggers the MDO-DWH Mapper to generate SQL statements for the, now non-generic, comparative cubes. After instantiating the comparative cubes, the rule family can be instantiated by binding of the concrete scores of the analysis situation to the generic scores defined by the generic rule family. This instantiation triggers the mapping of the rule family. The Mapper uses the instantiated cubes in order to compute the aggregated rule view. Note, that there are no views for generic rules, or an instantiation thereof, defined in the semDWH. Only the aggregated view of the instantiation of a generic judgement rule family, or two aggregated views in the case of the instantiation of a generic analysis rule family, are represented as views in the semDWH.

The aggregated view of a generic rule family instantiation has an equivalent structure and appearance as the aggregated view of a base rule family, and therefore can be treated equivalent during further rule evaluation.

4.5 Rule Evaluation

In this section we present the implementation of the rule evaluation process. First, we discuss the more straightforward evaluation of judgement rules, which also includes filtering for rule families that are relevant to a specific analysis situation. Then, we cover the evaluation of analysis rule families for both prerogative and presumed evaluation.

Rule evaluation is generally achieved by joining the view representing the analysis situation with the relevant aggregated rule views. The concrete evaluation of rules varies for the different rule types and parameters of the rule evaluation, however, the general process flow is rather similar as both rule types are evaluated on a specific analysis situation, that is, a comparative cube. This cube defines the facts for which rules are evaluated. The evaluation functions for the different rule types use this cube and the, depending on the type, explicitly or implicitly defined relevant rule families in order to compute a SQL DDL statement for a semDWH view, which contains the result of the rule evaluation. This statement is added to the mapping sequence like the mapping of a MDO concept. This mapping still has to be executed in the semDWH. Once the statement has been executed, the evaluation result can be retrieved by querying the semDWH for the rule evaluation view.

4.5.1 Judgement Rule Evaluation

In this subsection we discuss the implementation of judgement rule evaluation. We describe the filtering in order to identify applicable judgement rule families and show how to compute the result view for a specific judgement rule evaluation.

As defined in section 3.1 the evaluation of judgement rules occurs implicitly without specifying the specific judgement rule families to evaluate. To model this behaviour the rule engine provides a function that takes a comparative cube as input and returns the name of the judgement rule evaluation result for this cube. As each analysis query is modelled as cube this allows a seamless integration of the rule engine by routing all analysis situations through the provided evaluation function. The evaluation result can then be used for querying instead of querying the semDWH directly for the analysis situation.

Because of the implicit evaluation the first step of the rule evaluation is to compute a set of all judgement rule families suitable to the analysis situation at hand. An applicable judgement rule family is a rule family that is defined over facts contained in the analysis situation. Depending on the concrete situation it might be difficult to decide whether a specific analysis cube overlaps with a judgement rule family. Therefore, instead of identifying the overlapping rule families, the rule engine starts with a set of all rule families defined in the MDO and excludes those for which an overlap can be eliminated (see figure 4.3). This approach aligns with the employed prototyping approach as it allows to gradually improve the exclusion function in subsequent iteration steps.

Exclusion of unfitting rules is sound but incomplete, that is, all excluded rule families do not overlap with the analysis situation, but there might be some rule families that do not overlap with the analysis situation but are not excluded from the set of relevant rules. An advantage of this method of exclusion is that the usage of incomplete but sound reasoning of MDO concept relations does not lead to a different result compared to a complete and sound reasoning. The only difference is that in the case of incomplete reasoning more rule families than strictly necessary are evaluated against the analysis query without affecting the evaluation outcome. This is useful as the reasoning process currently implemented in the prototype system does not cover all different forms of concept definitions. For example, no reasoning can be computed for constructs that are defined in the MDO-DB by directly providing the concept's SQL representation.

As a tradeoff this incomplete exclusion algorithm can lead to an overhead in computation costs for rule evaluations as aggregated views might be computed and joined to an analysis result even though they do not contain any comparative facts in common. However, performance is not a focus of the prototype implementation and as the generation of aggregated judgement rule views and their evaluation did not lead to an apparent performance issue (see section 4.6) this behaviour is acceptable for the prototype implementation.

The exclusion function is implemented as a two-tier function that takes an analysis situation as input and returns a set of applicable judgement rule families for which an overlap cannot be ruled out. The first criteria for rule exclusion is the set of scores defined by the analysis situation. A rule family can be excluded from evaluation if a score of the rule family is not defined in the analysis situation. The second criteria is the comparative concept of the analysis situation in question. If the comparative concept of the analysis situation and of the rule family's root cube (the comparative cube of the root rule of the rule family) are disjoint, an overlap can be eliminated. The function for computing the relevant rule families is invoked during judgement rule evaluation, which in turn is triggered for each analysis situation whenever judgement rule evaluation is enabled, in line with the defined implicit evaluation of judgement rules.

To decide whether the scores of a given analysis situation and a rule family match it is necessary to define the matching behaviour for scores in the prototype implementation. Conceptually, a score is equal to another score, if both scores yield the exact same results for every fact under all circumstances. However, the semCockpit approach allows to define scores that yield the same results in different ways and in the current implementation it is not possible to decide whether two scores that are defined in different ways are conceptually equal. For the remainder of this work it

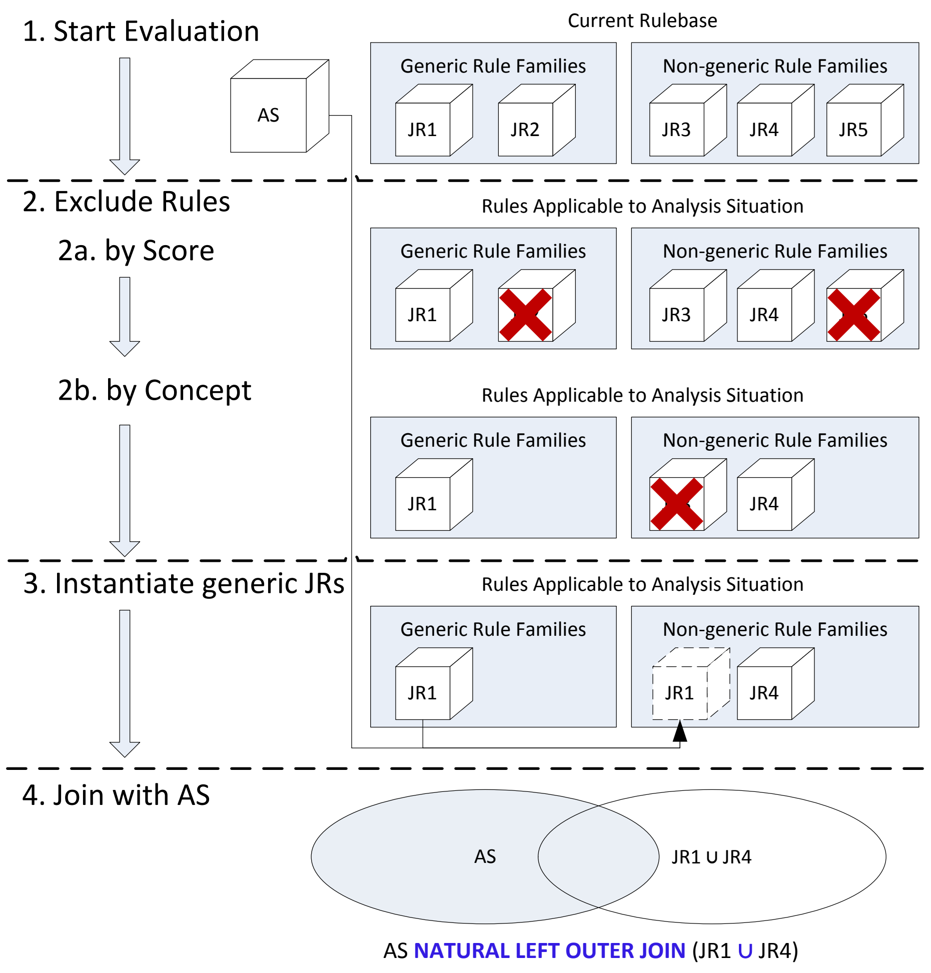
1. Start Evaluation
Current Rulebase
Generic Rule Families
Non-generic Rule Families
AS
JR1
JR2
JR3
JR4
JR5
2. Exclude Rules
2a. by Score
Rules Applicable to Analysis Situation
2b. by Concept
Rules Applicable to Analysis Situation
3. Instantiate generic JRs
Rules Applicable to Analysis Situation
4. Join with AS
AS
JR1 ∪ JR4
AS NATURAL LEFT OUTER JOIN (JR1 ∪ JR4)

Figure 4.3: Evaluation process for judgement rule families

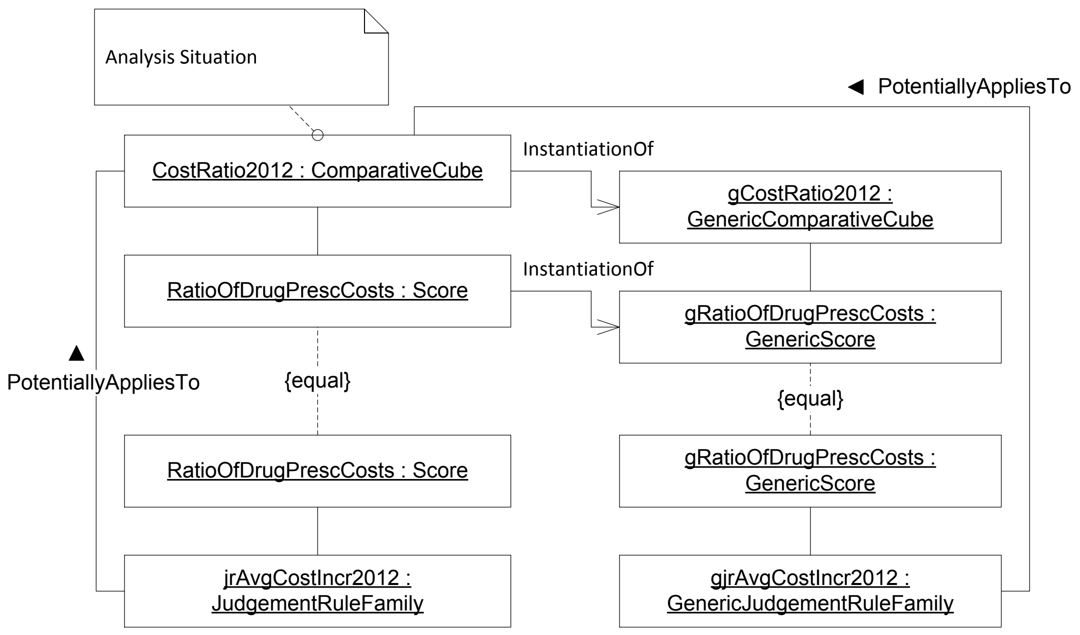
Analysis Situation
◀ PotentiallyAppliesTo
CostRatio2012 : ComparativeCube
InstantiationOf
gCostRatio2012 : GenericComparativeCube
RatioOfDrugPrescCosts : Score
InstantiationOf
gRatioOfDrugPrescCosts : GenericScore
▲ PotentiallyAppliesTo
{equal}
{equal}
RatioOfDrugPrescCosts : Score
gRatioOfDrugPrescCosts : GenericScore
jrAvgCostIncr2012 : JudgementRuleFamily
gjrAvgCostIncr2012 : GenericJudgementRuleFamily

Figure 4.4: Matching of scores between analysis situation and judgement rule family

is assumed, with one explicit exception, that a score is only equal to another score, if they share the same MDO identifier. The exception of this rule is the handling of generic scores and the non-generic scores generated by instantiating generic scores with concrete qualifiers. Since there is no doubt that two scores instantiating the same generic score with the same qualifiers are conceptually equal, these two scores are treated as equal.

For non-generic judgement rule families the exclusion based on the scores of the analysis situation is implemented as follows: First, the two sets of scores in question, one for the analysis situation and one for the judgement rule family, are retrieved. Then, each score of the analysis situation is compared to the scores of the judgement rule family in order to find matching scores (see figure 4.4). If for all scores of the rule family a matching score has been found, then the rule family cannot be excluded based on its scores. Note, that the analysis situation can define more scores than the rule family without leading to exclusion as the additional scores are simply ignored during rule evaluation. However, if there are one or more scores defined by the rule family that have no matching score in the analysis situation the rule has to be excluded. If such a rule family would not be excluded, its judgements would be irritating or even misleading as they are, at least partially, based on score values that are not part of the analysis situation.

Generic judgement rule families need a slightly more sophisticated score matching as the generic comparative cubes of generic rules can additionally define score qualifier domains. The score matching starts with the set of concrete scores defined by the analysis situation (see figure 4.4). The first step in deciding whether the scores match is to find for each generic score defined by the rule family a corresponding non-generic score in the analysis situation that is defined as an instantiation of the rule families generic score. If one or more generic scores of the rule family are not represented by a score instantiation in the analysis situation, the rule family can be excluded from evaluation. After finding those score pairs the qualifier domain restrictions of the comparative cube of the rule family's root rule have to be analysed. Only if the domain restrictions are not violated, the scores match. Qualifier domain restrictions are evaluated by checking whether the concrete qualifiers used for the score instantiation in the analysis situation are consistent with the qualifier domain concept defined by the generic comparative cube. If either a generic score of the rule family is not instantiated by the analysis situation or a generic score is instantiated with qualifiers that are not consistent with the qualifier domain of the root rule, then the generic rule family can be excluded from the list of relevant rules.

If the scores of a rule family and the analysis situation match, the comparative concept of the comparative analysis situation and the root rule of the rule family are compared. Checking the comparative concepts is easy to achieve when the needed information is provided, for example, through the reasoning component. First, the root rule of the rule family in question is selected using the hierarchy relations. Second, the comparative concept of the comparative cube of the rule as well as the comparative concept of the analysis situation are selected. Finally, it is evaluated whether those two concepts are disjoint using a specific disjoint relation, which is populated by the reasoning component and contains information about the disjointness of MDO concepts. If there is no entry in the relation comprising the two comparative concepts, then it is not certain that they are disjoint in alignment with the sound but incomplete reasoning approach employed. If there is an entry consisting of the two concepts in question, an overlap of the comparative points, and therefore of analysis situation and rule family can be eliminated. The exclusion function for generic and non-generic rule families based on the comparative concept only differs with regard to the relations in which the rule information is stored within the MDO-DB (see section 4.2). A generic comparative cube defines a comparative concept just like a non-generic comparative cube, and therefore the check can be conducted analogous.

Only rule families with matching scores that do not have a disjoint comparative concept for their root rule definition are added to the result set of the exclusion function, which contains the applicable judgement rule families for the current analysis situation. Due to the different structure of generic and non-generic rule families there are two separate exclusion functions, one computing the set of relevant generic rule families and one computing the set of relevant non-generic rule families.

As a result of the exclusion functions two sets of applicable judgement rule families, one containing generic and one containing non-generic families, remain, which now contain the rule families that are evaluated against the analysis situation. As described in section 4.4, the aggregated views of rule families are computed only when needed. Before computing the rule evaluation results, for each rule family in the sets of applicable rules, the SQL statement for its aggregated rule view has to be computed. The significance of the previously described exclusion process is that those statements are only needed for the remaining families for which an applicability could not be precluded. The computation of the aggregated view statements as well as the process of instantiating the generic rule families has already been described in section 4.4.

goi_time	goi_ins/goc_ins	goc_time	Ratio	Rule	Judgement
2012	Styria	2011	0.99	(null)	(null)
2012	Leoben	2011	1.04	jrAvgCostIncr2012	There is on...
2012	Lower Austria	2011	1.30	jrAvgCostIncr2012	There is on...
2012	Melk	2011	0.97	(null)	(null)
2012	Korneuburg	2011	1.07	jrAvgCostIncr2012	There is on...
2012	Horn	2011	1.06	jrAvgCostIncr2012	There is on...
2012	Upper Austria	2011	1.30	jrAvgCostIncr2012UA	In Upper Austria...
2012	Linz-Stadt	2011	1.50	jrAvgCostIncr2012UA	In Upper Austria...
2012	Wels-Stadt	2011	1.30	jrAvgCostIncr2012UA	In Upper Austria...
2012	Eferding	2011	0.98	(null)	(null)
2012	Steyr-Stadt	2011	1.13	(null)	(null)

Table 4.6: Judgement rule evaluation result

After the mapping of the aggregated views has been completed for both generic and non-generic judgement rule families, the evaluation result can be computed by combining the aggregated views with the analysis situation. The result of the rule evaluation is a SQL DDL statement for the evaluation result view, which represents the rule evaluation in the semDWH. The rule evaluation consists of all facts covered by the original analysis situation. Additionally, for each fact for which one or more judgements applies, those judgements, together with the specific rule that caused the judgement, are added to the evaluation result. There are three basic outcomes for a single fact. First, if no judgement applies, then the fact is contained in the evaluation result showing no judgement and no rule that triggered. Second, if exactly one judgement applies, then that judgement together with the triggering rule is annotated. Third, if more then one judgement applies to a specific fact, either through different rule families or through overlapping rules within a single rule family, the fact is included multiple times in the evaluation result, whereby each occurrence is annotated with one judgement and the corresponding triggering rule.

Example 25 (Judgement Rule Evaluation Result). Table 4.6 shows a simplified excerpt of a judgement rule evaluation result. The table shows the evaluation results for an analysis of the comparative cube CostRatio2012 with a single applicable judgement rule family jrAvgCostIncr. As the values for goi_ins and goc_ins are equal based on the cube definition, they are shown in a single column. The indentation shows the roll-up relationships of districts and provinces. For example, the district Leoben rolls-up to province Styria. Likewise, the districts Melk, Korneuburg, and Horn roll-up to province Lower Austria. Additionally, dimension roles that are restricted to the dimension specific all-node are omitted. The comparative point for the province of Styria does not satisfy the rule condition defined by rule jrAvgCostIncr2012, and therefore no judgement is included in the evaluation result. Conversely, the defined judgement is annotated for district Leoben by providing the triggering rule jrAvgCostIncr.CostRatio2012 and the corresponding judgement text. For the province of UpperAustria and its districts the more specific rule jrAvgCostIncr2012UA, with the activation condition RatioOfDrugPrescCosts > 1.2, applies. Therefore, no judgement is provided for district Steyr-Stadt, as the condition of the specific rule is not satisfied.

4.5.2 Analysis Rule Evaluation

In this subsection the implementation of analysis rule evaluation is presented. We provide the interface specification for analysis rule evaluation and show the implementation of the two evaluation strategies, presumed evaluation and prerogative evaluation.

Analysis rule evaluation is triggered explicitly for a specified set of analysis rule families. Therefore, the process of identifying the appropriate rules is superfluous as the set of rule families that should be evaluated is provided as input to the evaluation function. As the current prototype does not implement a rule execution model, the evaluation is described based on reporting rules. If a rule family defines specific actions, these are additionally annotated in the evaluation result but do no lead to any further actions or executions. The differences between the implementation of the evaluation strategies will be discussed later on in this section. First, the roles of the other input parameters of the evaluation function are examined.

The input required by the analysis rule evaluation function are: (1) an analysis situation in the form of a concrete comparative cube; (2) a list of granularities, which define the evaluation path; (3) the desired evaluation strategy; and (4) a non-empty set of analysis rule families. The comparative cube defines the scope of the analysis rule evaluation. An ordered list of granularities provides the hierarchical order used for evaluation. It is assumed that the granularity list is consistent in the way that it contains valid granularity definitions that form a roll-up hierarchy and are ordered from coarser to finer granularities. A set of analysis rule families, generic and non-generic, defines the set of rules that are evaluated in the analysis rule evaluation process.

Independent of the chosen evaluation strategy the rule evaluation starts with the generation of the aggregated rule view statements for non-generic analysis rule families and the instantiation and mapping for generic analysis rule families. For the instantiation of generic analysis rule families the concrete scores, respectively the qualifiers used for the definition of those scores, of the provided analysis situation are used. The instantiation of a generic analysis rule family uses the same principles as instantiation of a generic judgement rule family. First, it is checked if for each generic score of the rule family a corresponding instantiated score is defined by the analysis situation. Second, the generic comparative cube of each generic rule within the family is instantiated with the concrete qualifiers if the qualifiers are consistent with the qualifier domain of the respective cube. Third, a non-generic analysis rule family by generic instantiation is created based on the generic analysis rule and the non-generic scores of the analysis situation, which triggers the mapping for the aggregated rule views. Remember, that an analysis rule family defines two rule views, one constructed using the positive activation conditions, and one using the negative activation conditions.

After the instantiation of the required generic cubes and generic rule families is completed, the evaluation differs depending on the selected evaluation strategy. The differences in the behaviour and application of the implemented evaluation strategies are discussed in detail in section 3.2. As for a judgement rule evaluation the result of an analysis rule evaluation is a SQL DDL statement that has to be executed in the semDWH. The created evaluation object represents the results of the analysis rule evaluation as view over the data stored in the semDWH.

In order to compute the evaluation within a single SQL construct several nested subqueries are used. Subqueries are needed independent of the used evaluation strategy. First, a subquery is defined, which augments the initial comparative cube representing the analysis situation with columns for the level and roll-up nodes of each fact. Then, a subquery is generated for each granularity of the defined evaluation path by restricting the cube created in the first step to the specified granularity levels. Further actions depend on the evaluation strategy and are computed for each analysis rule family that is part of the rule evaluation.

For the prerogative strategy, positive activations are reported in the result, negative activations lead to no action and facts without activation are evaluated on the next lower granularity. Starting at the coarsest granularity two subsets are computed: First, the set containing facts with positive activation (`positive1`) is the result of a join between the subcube of the current granularity with the positive aggregated rule view. Second, the set of undecided facts (`undecided1`) is computed by subtract-

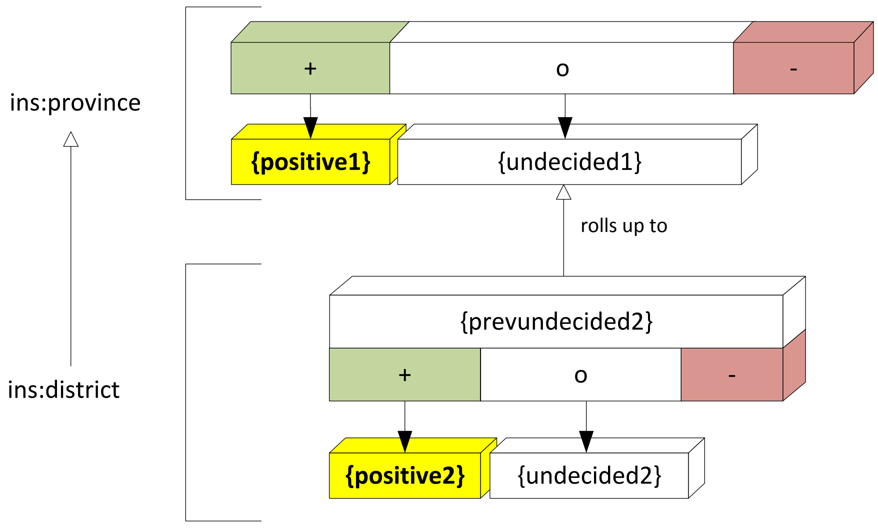

Figure 4.5: Fragmentation of prerogative analysis rule evaluation

goi_time	goi_ins/goc_ins	goc_time	Ratio	Rule	Act.	Report
2012	Styria	2011	0.99	arCostRatio2012	-	
2012	Leoben	2011	1.04	not evaluated		
2012	**Lower Austria**	**2011**	**1.30**	**arCostRatio2012**	+	x
2012	Melk	2011	0.97	not evaluated		
2012	Korneuburg	2011	1.07	not evaluated		
2012	Horn	2011	1.06	not evaluated		
2012	Upper Austria	2011	1.30	arCostRatio2012UA	o	
2012	**Linz-Stadt**	**2011**	**1.50**	**arCostRatio2012UA**	+	x
2012	Wels-Stadt	2011	1.30	arCostRatio2012UA	o	
2012	Eferding	2011	0.98	arCostRatio2012UA	-	

Table 4.7: Example set for prerogative analysis rule evaluation of rule family arCostRatio along dimension role insurant

ing facts contained in either the aggregated positive or negative rule view from the subcube of the granularity. For each further granularity the following subsets are computed in subqueries: First, the set containing the previously undecided facts (prevundecided2) is computed by selecting all facts of the current granularity that roll up to facts contained in the set of undecided facts of the previous granularity. Second, the set containing facts with positive activation for the current granularity (positive2) is computed by joining the positive aggregated rule view with the set of previously undecided facts. Finally, if there is a finer granularity defined for evaluation, the set of undecided facts on the current level (undecided2) has to be computed by subtracting the facts contained in either the positive or negative aggregated rule view from the set prevundecided2. Finally, the union of all subqueries containing facts with positive activation on the different granularities (positive1 and positive2) yields the desired evaluation outcome of a set of facts, which are reported and for which the action specified by the analysis rule family is recommended.

Example 26 (Prerogative Analysis Rule Evaluation Result). Table 4.7 shows the prerogative analysis rule evaluation for rule family arCostRatio along the insurant dimension on a sample set of comparative facts. Columns goi_ins and goc_ins are condensed to a single column and dimension roles that are restricted to their respective all-node are omitted. Indentation shows the roll-up relationships of districts and provinces. Highlighted tupels constitute the actual information contained in the evaluation result. The fact for the province of Styria satisfies the negative activation condition and is therefore not contained in the evaluation result. In addition, for Styrian districts, for example Leoben, the analysis rule is not evaluated. The province of Lower Austria satisfies the positive activation condition and is therefore added to set positive1. For Lower Austrian districts Melk, Korneuburg, and Horn, the analysis rule is not evaluated. The province of Upper Austria is contained in the set undecided1 as neither applicable activation condition is satisfied. On level district the set prevundecided2 therefore contains the Upper Austrian districts Linz-Stadt, Wels-Stadt, and Eferding, and for each district the analysis rule is evaluated. District Linz-Stadt is contained in set positive2, as it satisfies the positive activation condition. The union of the sets positive1 and positive2 constitutes the rule evaluation result. In the example, the result consists of the facts for the province of Lower Austria and the Upper Austrian district Linz-Stadt.

By using the presumed evaluation strategy positive activations are reported for all facts that are undecided along their roll-up path and if either a positive or a negative activation occurs at a specific granularity, then on finer granularities only those facts with opposite activation are reported. Therefore, starting at the first granularity two sets of facts are computed. The first set contains the facts matching the positive activation condition (rpos1) and the second one those that do not match the positive activation (lastneg1) and therefore are either undecided or match the negative activation condition. Facts that satisfy the positive activation condition on the coarsest granularity are also reported and added to a separate set lastpos1 for further evaluation. Evaluation on the next finer granularity depends on the results of the previous granularity level. The facts on the current granularity are divided into two sets, one containing all facts that roll up to a fact with positive activation, defined as rolluppos2 and one set containing the facts that roll up to an undecided fact or a fact with negative activation, defined as rollupneg2. For facts in the set rolluppos2, the facts satisfying the negative activation condition (rneg2) are reported, as those facts contradict the previously reported positive activation for their roll-up facts. Analogous, for the set rollupneg2 the facts matching the positive activation condition (rpos2) are reported. On the next finer granularity the same distinction of facts into rolluppos and rollupneg is made. Note, however, that the evaluation result of previous granularities might or might not have changed. In order to compute the intended result two additional fact sets have to be computed. The set lastpos2 contains all facts with the last matching activation condition along their roll-up hierarchy being positive. Accordingly, the set lastneg2 contains all facts with the last matching condition being negative. The only exception to these rules is that all facts with neither activation condition along their whole roll-up hierarchy are also included in the set of lastneg2 as they can be treated identically during further evaluation. The evaluation result for a presumed analysis rule evaluation consists of the union of (1) the facts on the coarsest granularity that satisfy the positive activation condition (rpos1), (2) all facts contained in the defined rolluppos sets that satisfy the negative activation condition (rneg2) and (3) all facts contained in the defined rollupneg sets that satisfy the positive condition (rpos2). For each of these facts the specific rule responsible for activation and whether the activation was positive or negative is included in the evaluation result.

Example 27 (Presumed Analysis Rule Evaluation Result). Table 4.8 shows the analysis rule evaluation for rule family arCostRatio using the presumed evaluation strategy along the insurant dimension on a sample set of comparative facts. The fact for the province of Styria satisfies the negative activation condition and is therefore contained in the set lastneg1. Styrian districts Leoben, Murau, and Weiz are part of the set rollupneg2. Leoben and Weiz do not satisfy the positive activation condition and are added to set lastneg2. However, Murau satisfies the positive activation condition and is added to set rpos2 and reported in the evaluation result. Province Lower Austria is contained in sets rpos1 and lastpos1 as it satisfies the positive activation condition of rule arCostRatio2012. Consequently, the Lower Austrian districts Melk, Korneuburg, Horn, and Scheibbs are part of set rolluppos2 and are checked for a negative activation. District Melk satisfies the negative activation condition and is added to set rneg2. The province Upper Austria satisfies neither the positive nor the negative activation condition of rule arCostRatio2012UA. Therefore, Upper Austrian

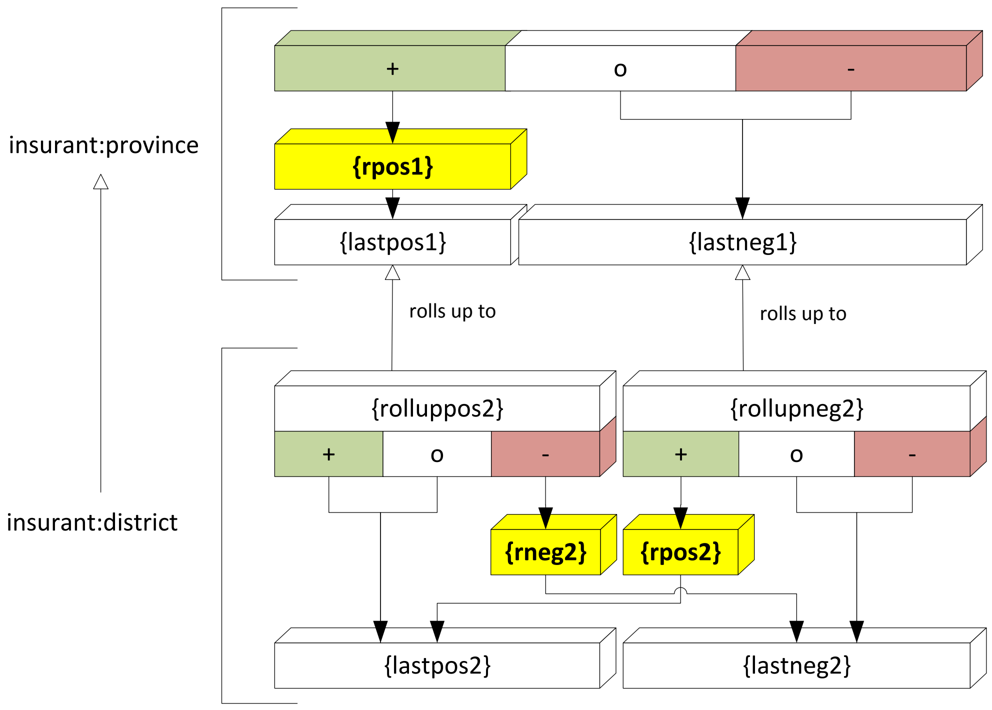

Figure 4.6: Fragmentation of presumed analysis rule evaluation

goi_time	goi_ins/goc_ins	goc_time	Ratio	Rule	Act.	Report
2012	Styria	2011	0.99	arCostRatio2012	-	
2012	Leoben	2011	1.04	arCostRatio2012	o	
2012	**Murau**	**2011**	**1.21**	**arCostRatio2012**	**+**	x
2012	Weiz	2011	0.89	arCostRatio2012	-	
2012	**Lower Austria**	**2011**	**1.30**	**arCostRatio2012**	**+**	x
2012	**Melk**	**2011**	**0.97**	**arCostRatio2012**	**-**	x
2012	Korneuburg	2011	1.07	arCostRatio2012	o	
2012	Horn	2011	1.06	arCostRatio2012	o	
2012	Scheibbs	2011	1.35	arCostRatio2012	+	
2012	Upper Austria	2011	1.30	arCostRatio2012UA	o	
2012	**Linz-Stadt**	**2011**	**1.50**	**arCostRatio2012UA**	**+**	x
2012	Wels-Stadt	2011	1.30	arCostRatio2012UA	o	
2012	Eferding	2011	0.98	arCostRatio2012UA	-	

Table 4.8: Example set for presumed analysis rule evaluation of rule family arCost-Ratio along dimension role insurant

districts are contained in set rollupneg2 and are evaluated against the analysis rule family in order to report districts that satisfy the positive activation condition. In the example, the district Linz-Stadt satisfies the positive activation condition and is therefore added to set rpos2. The final evaluation result for the example set contains the facts for Murau, Lower Austria and Linz-Stadt, which are reported as positive activation, and the fact for Melk, which is reported as negative activation as it contradicts the positive report for Lower Austria.

As stated before, the prototype implementation of the rule engine does not provide an action execution model for the actions defined by analysis rule families. The presented implementation is limited to report the affected facts according to the used evaluation strategy and annotate the stored action for reported facts with positive activation. Execution of the defined action with the appropriate parameters depending on the fact values together with an appropriate conflict resolution has to be conducted manually. Some additional remarks with respect to this topic are discussed in section 4.7.

Description of the rule evaluation for analysis rules is based on the assumption that the evaluation function is explicitly called. However, until now it has not been defined how this function call is triggered. In the most basic case the user directly calls the evaluation through some interface provided by the semCockpit frontend. Alternatively, cascading triggers can be used to start rule evaluations at specific events, for example, after new data has been loaded into the data warehouse or at specific times. Because of the limited implementation of an action execution model for analysis rules, the practical use for such triggers is very limited in the current prototype implementation.

4.6 Preliminary Performance Studies

In this section we discuss the results of several performance tests based on the current prototype implementation. As performance and performance optimisation was not specifically considered during prototype implementation, some deficiencies were observed during preliminary performance tests. We provide a summary of these tests in order to guide future work on performance optimisation by identifying the components of the implementation that impact performance the most. First, we provide information on the setup we used for the tests as well as the test method employed. Then, we present the performance results for the computation of rule mappings and the execution of mappings in the semDWH.

4.6.1 Test Setup

The performance tests have been conducted using Oracle Database 11.2g with the following test configuration. Both the MDO-DB and the semDWH were physically located on the same server in two different database schemata. All tests were conducted using a local database server running on a Microsoft Windows 7 (x64) machine with 4 GB RAM and a 2.20 GHz Intel Core 2 Duo CPU. The database server has been configured with an allocated memory of 1.4 GB RAM. The database has been accessed through Oracle SQL Developer.

The conducted performance measures are based on the total execution time of the tested transactions; other performance indicators such as hard drive accesses or memory usage were not measured. The execution times of invoked function and method calls have not been measured individually. Therefore, the measured execution times might contain considerable errors. However, a precise measuring of performance was neither necessary nor intended for this preliminary tests in order to achieve the intended insights. Test result values constitute the average execution time over multiple (at least five) test executions.

The main focus of the test method is to identify the components with the largest effects on overall performance. In order to do so, rough performance measurements are sufficient. For the most part, the presented tests are limited to the concepts that are used and have been introduced for rule definitions, that is, rules, rule families and rule evaluations. Other MDO concepts that are used in rule definitions, like comparative concepts and comparative cubes, are assumed to be already defined in the MDO-DB and in the semDWH if not stated otherwise.

4.6.2 Rule Mapping Performance

In this subsection we show the execution times for the mapping of different rule concepts. The rule mappings are independent of the data stored in the semDWH and solely depend on the objects that are stored in the MDO-DB.

In figure 4.7 the average execution times for the mapping of analysis and judgement rules are depicted. This test encompasses the insertion of a new rule definition expression in the MDO-DB, the in this way triggered mapping computation and the insert of the computed mapping into the *mapping sequence* where the statement is stored for later execution in the semDWH. The difference in the measured times is primarily explained by the fact that for a single analysis rule two mappings, one for positive evaluation and one for negative evaluation, have to be computed compared to a single mapping for judgement rules. Note, that the depicted mappings are for single rule views and not for the aggregated rule view of a rule family, which will be discussed later on. The execution time for computing mappings for single rules is independent of the rule's rule family or the ordering of the rule within the hierarchy

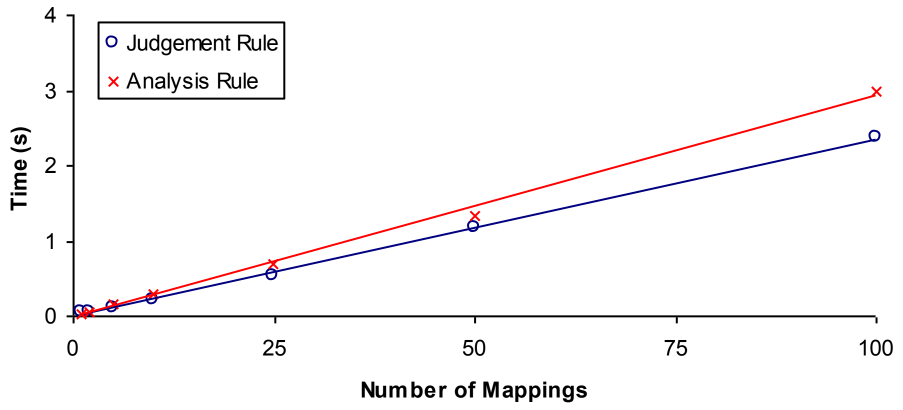

Figure 4.7: Mapping execution time for judgement and analysis rules based on number of rule mappings

of its rule family.

The mapping for single rules does only occur once at the time of the rule's definition and additionally whenever the entry of the rule's representation in the MDO is updated.

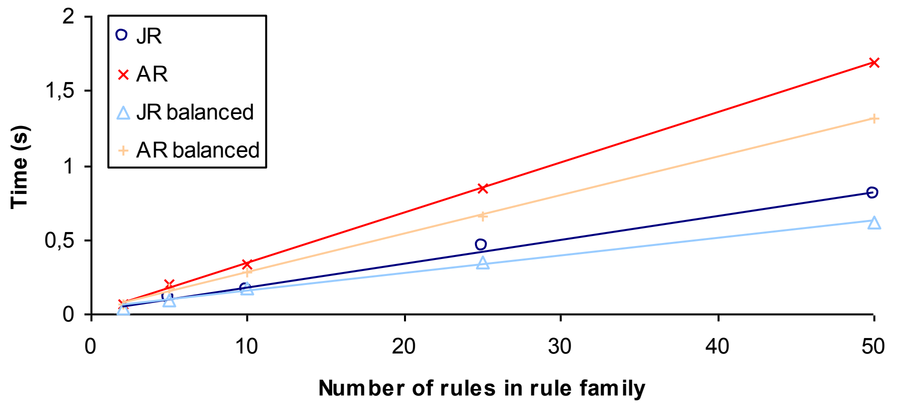

Figure 4.8: Mapping execution time for aggregated rule views for balanced and unbalanced rule hierarchies

Mapping of aggregated rule views depends on the derived rule hierarchies and the number of rules within the rule family. Figure 4.8 tries to illustrate these dependencies by showing the differences in the execution times for aggregated rule view mappings for rule families with different attributes. The figure shows the mapping execution at each data point for a single rule family. The diagram shows the linear relation of the mapping time compared to the number of rules within a given rule family. Additionally, figure 4.8 shows some differences between the mapping time depending on the structure of the family's rule hierarchy. The two specific cases represent (1) a linear hierarchy, that is, each rule (excluding top and bottom) defines exactly one direct superrule and one direct subrule and (2) a balanced case as a hierarchy in the form of a binary tree, where each rule defines one direct superrule and two direct subrules. Note, that the existence of the correct entries in the rule hierarchy tables are assumed to be already present in the MDO-DB. As in the case of single rules, the mapping computation is slower for analysis rule families as two aggregated mappings are computed.

The mapping time for aggregated rule views is important in the current implementation as they are computed during rule evaluation, for each applicable rule family in the case of judgement rule evaluation, and for each defined rule family in the case of analysis rule evaluation. Therefore, the mapping costs add directly to the performance costs of rule evaluation in contrast to the mappings for single rules, which are computed beforehand at the time of the rule's definition. This might be especially true for judgement rules, as the concept of implicit evaluation might lead to a high number of applicable rule families, and therefore to a high number of aggregated rule mappings. However, as most rule families probably do not have very complex hierarchies the estimated effect on rule evaluation time is not prohibitive.

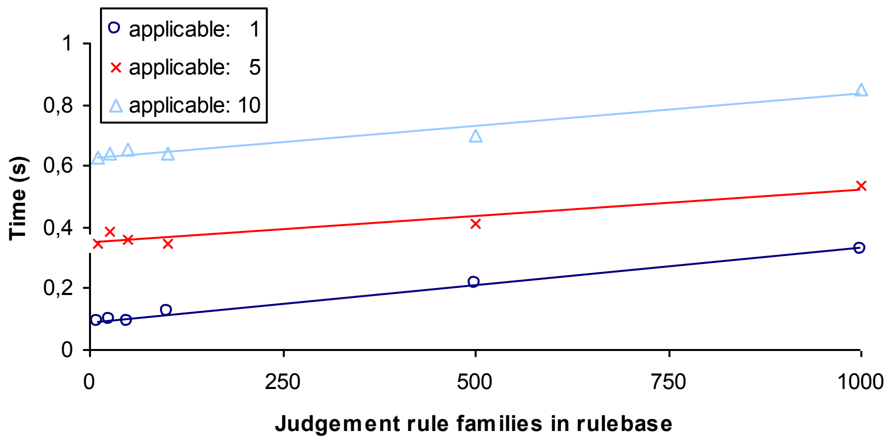

Figure 4.9: Execution time for filtering for applicable judgement rules and aggregated rule view mapping

Next, figure 4.9 shows the result of a test concerning the filter function responsible for filtering for applicable judgement rules. The plotted lines stand for different test settings with a different number of actually applicable rule families added in addition to the amount of not applicable rules in the rulebase. This test case encompasses the process of judgement rule evaluation in the MDO-DB and consists of the filtering of applicable rules, the computation of the aggregated rule views for applicable rules (note that all applicable rule families in the test case consisted of a single rule), and the mapping for the judgement rule evaluation view.

The results of this test show that the implemented filter function does scale well with an expanding rulebase.

The conducted tests regarding the mapping of rule constructs suggest, that the rule mapping performance is sufficient and currently does not require specific attention regarding performance optimisation. The rule mappings only seize the performance costs realised in the MDO-DB. Therefore, the next subsection focuses on performance costs in the semDWH.

4.6.3 Evaluation Performance in semDWH

This subsection discusses the performance of the generation of rule views and the querying of rule evaluations in the semDWH. The presented test results are based on a factitious data set containing 300.000 drug prescription facts for about 95.000 insurants.

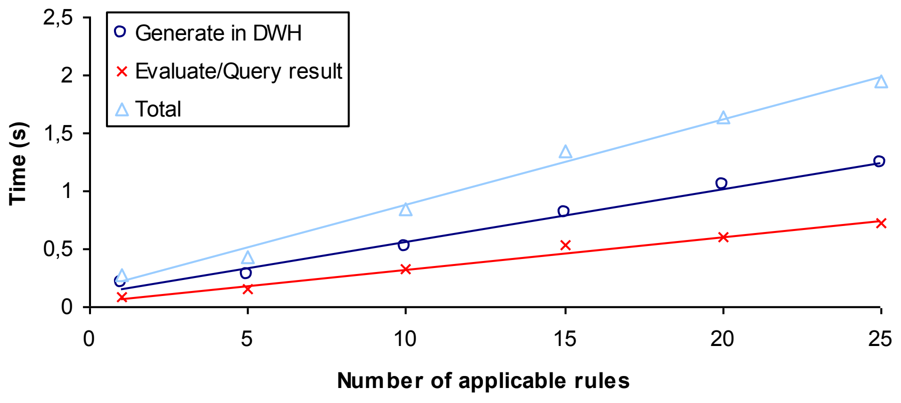

Figure 4.10: Execution time for judgement rule evaluation based on materialised comparative cubes

Figure 4.10 shows the required time for a judgement rule evaluation in the semDWH, by creating the previously mapped rules in the semDWH and querying the rule evaluation view for the results. It is important to note, that the depicted results are based on the fact that all comparative cubes that underlie the rule evaluation are already in existence and are available as materialised views in the semDWH.

The necessary calculations for the computation of comparative cubes as described in section 2.3 are by far the most expensive operation of the system prototype. As the rule engine implementation is built on top of the semCockpit system, which defines comparative cubes, optimising their computation is not within the scope of this work. Nevertheless, it is necessary to at least highlight this issue as it is not always easily possible to calculate and materialise comparative cubes prior to rule evaluation.

In the described test environment the computation of a single comparative cube, like the one used as illustrating example in section 2.3, takes around 40 minutes, and therefore dwarfs all other performance costs. The optimisation of the calculation of comparative cubes is still work in progress and some considerable improvements have already been achieved, which bring down the computation time for comparable cubes to somewhere between five to ten minutes. However, this is still prohibitively expensive for some use cases.

Note, that the test environment constitutes a rather minimal system in the context of data warehousing and that through the application of more powerful servers with more dedicated resources the computation times can be lowered significantly. Nevertheless, the data acquired during development suggest that further performance improvements through query optimisation and possibly more significant changes of the semCockpit prototype are necessary in order to achieve practicable results.

4.7 Discussion

In this section we discuss the presented implementation and consider possible extensions and enhancements as well as current limitations of the rule engine prototype by covering the topics of multi-user access, action execution, rule events and rulebase management. This section covers functionality that has to be implemented in order to allow for an useful application of the current prototype features in a production environment as an active database system. Limitations from a performance standpoint are not covered in this section, as we discussed the results of a preliminary performance study in detail in the previous section.

The presented prototype constitutes a first implementation of judgement and analysis rules in the context of comparative data analysis. Not only this, but additionally the implementation of rules are built in a way that allows rule definitions based on the concepts contained in a multi-dimensional ontology that explicitly defines business terms. We also showed, to our knowledge, the first implementation of the decision scope approach for data analysis. Still, several questions, which we did not elaborate on in this work, remain to be considered before the semCockpit system can be used in a production environment.

Further topics discussed in this section consist of the main shortcomings of the prototype implementation with regard to the features an active database management system should provide according to Act-Net Consortium [1996]. Note, that the aim of the prototype implementation was to show that the implementation of judgement and analysis rules on top of the semCockpit approach is possible with an off-the-shelf database management system and not to implement a full-fledged active database management system. Nevertheless, in order to fully use the implemented concepts, some form of active database system is needed. Therefore, the features defined in Act-Net Consortium [1996] provide a useful guideline for future development of the rule engine prototype.

One limitation of the presented system is the lack of multi-user functionality. In the current state the prototype system does not provide a sophisticated support for multiple users. The multi-user capabilities therefore are limited to the general functions provided by the Oracle database system, which do not suffice in the context of the prototype application. Multi-user capabilities would encompass functionalities in order to define rules that are visible only for certain users and user groups as well as some additional consistency checks in order to assert a valid state for the MDO-DB across multiple users. As multi-user functionality was not within the scope of the first prototype implementation, additional research is necessary in order to identify necessary implementation steps.

As mentioned before, a main limitation of the presented prototype in terms of functionality is the lack of an appropriate action execution model that allows to automatically execute actions based on the result of analysis rule evaluations. The action of an analysis rule should act on objects that are available in the data warehouse and in OLTP systems [Thalhammer and Schrefl, 2002, p. 1204]. However, the semCockpit system prototype does not contain an OLTP system as the implementation of automatic action execution was not in the scope of the presented research. Thalhammer and Schrefl [2002] and Thalhammer et al. [2001] contain considerable thoughts on the functionality needed to support such an action execution feature. Two essential features that are not contained in the current rule engine prototype concern (1) a more precise definition of the action of a rule, possibly by defining a primary dimension level, and (2) some functionality for detection and resolution of conflicting decisions. The primary dimension level is a particular dimension level, which determines the application domain of the rule's action. In this approach, the action represents a transaction in an OLTP system as method for instances of the primary dimension level [Thalhammer and Schrefl, 2002, p. 1198]. If the rule engine should support such actions, then the definition of the primary dimension level for actions is necessary. The application of actions on an instance level additionally requires some functionality for conflict resolution as different analysis rule evaluations can lead to conflicting actions for the same instance. Thalhammer et al. [2001, p. 255] propose to handle such conflicts by defining conflict resolution tables, which specify for each pair of conflicting methods a corresponding conflict resolution method that will be executed in the OLTP system. Implementation of the proposed action execution system is one of the essential features that would be expected from a complete active data warehouse system.

The definition of events that lead to the execution of analysis rules has not been discussed thus far in detail. Due to their conceptualisation and implicit evaluation, judgement rule families are not triggered by any event other than the execution of an analysis situation with activated judgement rule evaluation. In contrast, analysis rules should provide some event functionality as they define specific actions to be executed based on the evaluation result. The current prototype does not define explicit events for analysis rules. In the current implementation state, triggers within Oracle database can be used in order to model events leading to predefined analysis rule evaluations. However, such triggers would have to contain a lot of information buried within their definition. Based on this triggers alone it would be hard for business analysts to get an overview on the defined triggering events for a set of rules that are active at a given point in time. Therefore, the maintenance and evolution of the rulebase would be exceptionally time-consuming and error-prone. Some guidelines and thoughts on how to implement a dedicated event model for analysis rules, which would mitigate these problems, can be found in Thalhammer and Schrefl [2002] and Thalhammer et al. [2001].

An additional topic constitutes the adequacy of the prototype with respect to rulebase management and rulebase evolution. In order to provide a full rulebase management, the following features have to be considered [Act-Net Consortium, 1996, p. 43]: (1) Information about the current rules within the database can be retrieved at any time. (2) The rulebase is changeable over time, which includes the definition of new rules as well as the removal of others. (3) It is possible to modify the event, condition and action definitions of existing rules and (4) rules can be disabled and enabled. Only the first feature is fully supported by the current implementation as the definition expressions of rules are stored within the MDO-DB and can be retrieved at any time. It is also possible to add new rules to the rulebase by inserting new rule definition expressions into the database. The removal of rules and rule families from the rulebase is not fully implemented but would require minor effort. In order to remove a single rule from the rulebase, the rule's entry in the MDO-DB has to be deleted. Additionally, the rule hierarchy for the rule's rule family has to be updated and the appropriate delete statement for the rule view has to be generated and stored in the mapping sequence. The removal of a whole rule family requires the deletion of all of the rules in the rule family as well as the entries for the rule family from the MDO-DB. As in the case of removing a single rule from the rulebase a delete statement for all of the affected rule views and the aggregated rule family view has to be generated. Changes within rules are not actively supported in the current implementation but if the entries of the rule tables, which store the rule definitions are altered, the computation of a new mapping for the affected rule is triggered. However, the rule condition tables are not covered by triggers, and therefore changes within these relations go unnoticed and are not transferred to the semDWH representation of the rule. Enabling and disabling of rules and rule families is currently not implemented. This feature might be modeled by an additional relation that stores the activation state for each rule and rule family. During rule evaluation an additional check would verify that the rule is active and would remove the rule from evaluation if it is not enabled independent of the event that lead to the rule evaluation.

In the current state of the prototype implementation considerable limitations exist as described in this section. From a functional point of view, these limitations are explained by the fact that the system constitutes a research prototype.

5 Summary and Outlook

In this thesis we have presented the prototype implementation of a system that allows to define and evaluate rules as concepts of a multi-dimensional ontology. These rules are specifically designed for comparative data analysis in the context of data warehousing. The rule engine is part of the semCockpit system prototype, which defines a multi-dimensional ontology for explicitly capturing business terms in a central repository and provides an integrated approach towards more comprehensible analyses. Formalised business terms can be used for the definition of judgement and analysis rules.

The semCockpit approach underlying the rule engine implementation includes both the concept and structure of a semCockpit data warehouse and the concept of a multi-dimensional ontology, which enriches the data warehouse with a set of explicitly defined concepts. In the context of the MDO different concept types, namely entity concepts, dimensional concepts, multi-dimensional concepts, comparative concepts and multi-dimensional metaconcepts can be defined. Measures quantify facts of interest, and scores relate measure values of a group of interest to a group of comparison. Generic measures and scores are defined by using generic qualifiers, which are substituted with concrete concepts for instantiation. Cubes and comparative cubes are used to define a measure or score application and represent analysis situations. Generic comparative cubes can be used to model a generic analysis situation. The MDO-DWH Mapper handles the communication between the MDO and the semDWH and translates MDO construct definitions to executable SQL statements.

Judgement and analysis rules can be defined based on MDO concepts. Judgement rules are used to annotate previously gained insights to comparative data analysis situations. Therefore, otherwise tacit knowledge can be defined explicitly. Analysis rules can be used for the hierarchical analysis of multi-dimensional comparative data and allow for different evaluation strategies in order to prevent information overload and obtain relevant information depending on the current use case. Rules are organised in hierarchies within rule families, which allow to apply overriding in the sense that for each point only the most specific rule within a rule family is evaluated. Generic rules are used to define rules based on generic comparative analysis situations in order to define rules that are valid over a broad range of similar comparisons, and therefore reduce the number of rules that have to be defined.

We have shown that the established concepts can be implemented with an off-the-shelf database management system by implementing a research prototype using Oracle database. The motivation of this work was to show that an implementation of the rule engine is feasible with a conventional DBMS. A prototype covering the base functionality for the realisation of the rule engine has been developed and evaluated.

Functional limitations of the current prototype implementation include the lack of multi-user access, a rule execution model for automatic execution of analysis rule actions, a defined event model and shortcomings in the flexibility of rulebase management.

Preliminary performance measurements suggest, that considerable work has to be done in order to improve the performance of cube calculations to a satisfiable level. However, the data suggests, that the performance costs incurred by operations of the rule engine are negligible compared to the performance costs of creating and instantiating comparative cubes. The implementation of comparative cubes is part of the semCockpit prototype but does not belong to the rule engine component and was therefore not within the scope of this work.

From a conceptual standpoint, one promising extension of the rule engine constitutes the implementation of guidance rules in combination with analysis graphs as envisioned by Neuböck et al. [2013]. An analysis graph is the model of an analysis process and comprises multiple analysis situations and their relationship in terms of navigation steps. Guidance rules are used to guide the business analyst during data analysis. Guidance rules define depending on the current analysis situation and the measure and score values, recommended, potentially promising navigation steps for the analyst to follow in subsequent analyses. A similar behaviour can be modeled using the structure of analysis rules as defined in this thesis and combine them to an evaluation similar to the described implicit evaluation of judgement rules. Some work has to be done in order to define and model the action execution for guidance rules in a way that allows the execution of the desired navigation steps.

Bibliography

Act-Net Consortium, C. (1996). The active database management system manifesto: A rulebase of adbms features. *SIGMOD Rec.*, 25(3):40–49.

Agrawal, R., Gupta, A., and Sarawagi, S. (1997). Modeling multidimensional databases. In *Proceedings of the Thirteenth International Conference on Data Engineering*, ICDE '97, pages 232–243, Washington, DC, USA. IEEE Computer Society.

Araque, F. (2003). Real-time data warehousing with temporal requirements. In *CAiSE Workshops*.

Bouattour, S., Messaoud, R. B., Boussaid, O., Abdallah, H. B., and Feki, J. (2009). A framework for active data warehouses. In *The 2008 International Arab Conference on Information Technology (ACIT'2008), Hammamet, Tunisie.*

Browne, D., Desmeijter, B., Dumon, R. F., Kamal, A., Leahy, J., Masson, S., Rusak, K., Yamamoto, S., and Keen, M. (2010). Ibm cognos business intelligence v10.1 handbook.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., and Savo, D. F. (2011). The mastro system for ontology-based data access. *Semant. web*, 2(1):43–53.

Datta, A. and Thomas, H. (1999). The cube data model: A conceptual model and algebra for on-line analytical processing in data warehouses. *Decis. Support Syst.*, 27(3):289–301.

Golfarelli, M., Maio, D., and Rizzi, S. (1998). The dimensional fact model: A conceptual model for data warehouses. *International Journal of Cooperative Information Systems*, 7:215–247.

Golfarelli, M., Rizzi, S., and Cella, I. (2004). Beyond data warehousing: What's next in business intelligence? In *Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP*, DOLAP '04, pages 1–6, New York, NY, USA. ACM.

Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. (1997). Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov.*, 1(1):29–53.

Greenwald, R., Stackowiak, R., and Stern, J. (2007). *Oracle Essentials: Oracle Database 11g*. O'Reilly & Associates, Inc.

Inmon, W. H. (1992). *Building the Data Warehouse*. John Wiley & Sons, Inc., New York, NY, USA.

Khouri, S. and Ladjel, B. (2010). A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In *Proceedings of the ACM 13th International Workshop on Data Warehousing and OLAP*, DOLAP '10, pages 19–24, New York, NY, USA. ACM.

Kimball, R. and Strehlo, K. (1994). Why decision support fails and how to fix it. *Datamation*, 40(11):40–43.

Lim, L., Wang, H., and Wang, M. (2007). Unifying data and domain knowledge using virtual views. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 255–266. VLDB Endowment.

McCarthy, D. and Dayal, U. (1989). The architecture of an active database management system. *SIGMOD Rec.*, 18(2):215–224.

Mohania, M. and Narang, I. (2003). Policy based enterprise (active) information integration. In Mařík, V., Retschitzegger, W., and Štěpánková, O., editors, *Database and Expert Systems Applications*, volume 2736 of *Lecture Notes in Computer Science*, pages 1–7. Springer Berlin Heidelberg.

Nebot, V., Berlanga, R., Pérez, J., Aramburu, M., and Pedersen, T. (2009). Multidimensional integrated ontologies: A framework for designing semantic data warehouses. In Spaccapietra, S., Zimányi, E., and Song, I.-Y., editors, *Journal on Data Semantics XIII*, volume 5530 of *Lecture Notes in Computer Science*, pages 1–36. Springer Berlin Heidelberg.

Nebot, V. and Llavori, R. B. (2012). Building data warehouses with semantic web data. *Decision Support Systems*, 52(4):853–868.

Neuböck, T., Neumayr, B., Rossgatterer, T., Anderlik, S., and Schrefl, M. (2012). Multi-dimensional navigation modeling using bi analysis graphs. In *Proceedings of the 2012 International Conference on Advances in Conceptual Modeling*, ER'12, pages 162–171, Berlin, Heidelberg. Springer-Verlag.

Neuböck, T., Neumayr, B., Schrefl, M., and Schütz, C. (2013). Ontology-driven business intelligence for comparative data analysis. In Zimányi [2014], pages 77–120.

Neumayr, B., Schrefl, M., and Linner, K. (2011). Semantic cockpit: An ontology-driven, interactive business intelligence tool for comparative data analysis. In Troyer, O., Bauzer Medeiros, C., Billen, R., Hallot, P., Simitsis, A., and Mingroot, H., editors, *Advances in Conceptual Modeling. Recent Developments and New Directions*, volume 6999 of *Lecture Notes in Computer Science*, pages 55–64. Springer Berlin Heidelberg.

Neumayr, B., Schrefl, M., and Schütz, C. (2013). Semantic enrichment of data cubes: Multi-dimensional ontologies and their representation in sql and owl. In *Proceedings of the 12th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2013)*, Graz, Austria. Springer.

Romero, O. and Abelló, A. (2007). Automating multidimensional design from ontologies. In *Proceedings of the ACM Tenth International Workshop on Data Warehousing and OLAP*, DOLAP '07, pages 1–8, New York, NY, USA. ACM.

Romero, O. and Abelló, A. (2013). Open access semantic aware business intelligence. In Zimányi [2014], pages 121–149.

Schrefl, M., Neumayr, B., and Stumptner, M. (2013). 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)*, Adelaide, South Australia.

Sciarrone, F., Starace, P., and Federicit, T. (2009). A business intelligence process to support information retrieval in an ontology-based environment. In *Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on*, pages 896–901.

Spahn, M., Kleb, J., Grimm, S., and Scheidl, S. (2008). Supporting business intelligence by providing ontology-based end-user information self-service. In *Proceedings of the First International Workshop on Ontology-supported Business Intelligence*, OBI '08, pages 10:1–10:12, New York, NY, USA. ACM.

Thalhammer, T. and Schrefl, M. (2002). Realizing active data warehouses with off-the-shelf database technology. *Software: Practice and Experience*, 32(12):1193 – 1222.

Thalhammer, T., Schrefl, M., and Mohania, M. K. (2001). Active data warehouses: Complementing olap with active rules. *Data Knowledge Engineering*, 39(3):241 – 269.

Zimányi, E., editor (2014). *Business Intelligence - Third European Summer School, eBISS 2013, Dagstuhl Castle, Germany, July 7-12, 2013, Tutorial Lectures*, volume 172 of *Lecture Notes in Business Information Processing*. Springer.

Zwick, M., Lettner, C., and Hawel, C. (2007). Implementing automated analyses in an active data warehouse environment using workflow technology. In *Proceedings of the 2Nd International Conference on Trends in Enterprise Application Architecture*, TEAA'06, pages 341–354, Berlin, Heidelberg. Springer-Verlag.

List of Figures

List of Tables