Ontology-Driven Business Intelligence for Comparative Data Analysis

Thomas Neuböck[1], Bernd Neumayr[2], Michael Schrefl[2(✉)], and Christoph Schütz[2]

[1] Solvistas GmbH, Graben 18, 4020 Linz, Austria
thomas.neuboeck@solvistas.at
[2] Johannes Kepler University Linz, Altenberger Straße 69, 4040 Linz, Austria
{neumayr,schrefl,schuetz}@dke.uni-linz.ac.at

Abstract. In this tutorial, we present an ontology-driven business intelligence approach for comparative data analysis which has been developed in a joint research project, *Semantic Cockpit* (semCockpit), of academia, industry, and prospective users from public health insurers. In order to gain new insights into their businesses, companies perform comparative data analysis by detecting striking differences between different, yet similar, groups of data. These data groups consist of measure values which quantify real-world facts. Scores compare the measure values of different data groups. semCockpit employs techniques from knowledge-based systems, ontology engineering, and data warehousing in order to support business analysts in their analysis tasks. Concept definitions complement dimensions and facts by capturing relevant business terms which are used in the definition of measures and scores. Furthermore, domain ontologies serve as semantic dimensions and judgement rules externalize previous insights. Finally, we sketch a vision of analysis graphs and associated guidance rules to represent analysis processes.

Keywords: Business intelligence · OLAP · Data warehouses · Semantic technologies

1 Introduction

For their analysis tasks, business analysts rely on a data warehouse which organizes data as multi-dimensional facts. Each fact represents a business event that has been recorded in a transactional database. In the data warehouse, facts are identified by dimensions and quantified by measures. For example, the recipient patient, the issuing doctor, and the date of issuance identify a drug prescription. The prescribed quantity and the incurred costs are measures which quantify the drug prescription.

Business intelligence (BI) tools support interactive reporting over the corporate data warehouse through online analytical processing (OLAP). OLAP operations allow for the selection of different data groups and the aggregation

E. Zimányi (Ed.): eBISS 2013, LNBIP 172, pp. 77–120, 2014.
DOI: 10.1007/978-3-319-05461-2_3,

of measures. Business analysts employ OLAP operations for different types of analysis. First, *is-reporting* provides summary or detail information about a current or past business situation. For example, a health insurance manager might be interested in the total costs of drug prescriptions from last month. Second, *is-to-target comparison* contrasts a current or past business situation with a (hypothetical) target situation. For example, a health insurance company sets out a target figure for the monthly costs of drug prescriptions which a business analyst contrasts with the actually incurred costs from last month. Third, *is-to-is comparison* contrasts different, yet similar, business situations in order to gain insights into the analysis area. For example, a business analyst compares the incurred costs of drug prescriptions for different groups of patients in various months. In this tutorial, we focus on is-to-is comparison of data.

Whereas the tasks of is-reporting and is-to-target comparison tend to be simple and structured, is-to-is comparison is fundamentally more complex and often left to human intuition. The business analyst faces the challenge of defining meaningful comparisons. This definition demands knowledge about relevant business terms as well as their semantics. The business analyst must select relevant comparison groups, identify the subsets of facts to consider, and define the relevant measures for the illustration of the differences between the comparison groups. For example, a health insurance manager might be interested whether there are any exceptional differences between any groups of diabetes mellitus patients. The business analyst identifies patients from different insurance companies, provinces, and of different age as meaningful comparison groups. The business analyst considers only the facts that concern oral anti-diabetic drugs, insulin, and metformin. For the illustration of the differences between groups, the business analyst defines the average drug costs per patient and the drug costs that are prescribed by general practitioners for regular patients.

Comparative data analysis is an interactive, exploratory, and iterative process which employs OLAP for is-to-is comparison. Initially, relevant comparison groups and the measures for the illustration of the differences between the comparison groups are unknown. As a first step, the business analyst determines comparison groups and measures. Once comparison groups and measures are determined, the business analyst repeatedly conducts the comparative analysis with varying parameters. By varying the parameters of the analysis, the business analyst discovers dependencies among the data. Thereby comparative data analysis leads to the detection of exceptional differences between selected groups of data, suggests plausible explanations as well as cause-and-effect relationships, and highlights paths for further investigation. In this sense, comparative data analysis is not a replacement for data mining. Rather, comparative data analysis precedes data mining, assisting business analysts in the formulation of appropriate questions to statisticians.

Traditional BI tools fail to support the full process of comparative data analysis. The definition of business terms is left to the business analyst rather than providing a central repository which unambiguously defines the semantics of business terms. The comparison of data is a simple enumeration of selected measures, their interpretation left to the business analyst. In traditional BI tools, e.g.,

Tableau[1] and Oracle BI[2], business analysts conduct analysis in an ad-hoc manner with each analysis task started from scratch. The *Semantic Cockpit* (semCockpit) approach as presented herein fully supports comparative data analysis.

In semCockpit, a multi-dimensional ontology (MDO) provides an unambiguous definition of business terms for the specific needs of OLAP. Business terms are hierarchically ordered and become first-class citizens, which allows analysts to employ business terms in formulating OLAP queries. Furthermore, semantic dimensions allow for the integration of existing domain ontologies in OLAP.

Ontology-based measures and scores use concepts of ontologies to specify the data to be included in the calculation of derived measures and scores. Scores make comparison a first-class citizen in semCockpit. They capture the results of a comparison explicitly and free the business analyst from visual comparison by diagram inspection. A generic definition facilitates the reuse of scores in various analysis situations to avoid vast enumerations of similar measures.

Analysis graphs and rules capture otherwise tacit knowledge about how to proceed in analysis and about possible explanations of analysis results. Analysis graphs explicitly define the process of the analysis. Analysis rules recommend the initiation of a specific analysis process to the business analyst when certain conditions are met. Guidance rules lead the business analyst through the analysis graph. Judgement rules provide explanations for exceptional values. Analysis, guidance, and judgement rules externalize actionable knowledge otherwise tacit to the business analyst.

Existing BI approaches employ ontologies complementary to the semCockpit approach. Ontology-based data warehouse design [20,29,31,38] employs ontologies to automate tasks concerning construction of data warehouses and ETL processes. Ontologies serve as foundation for open access semantic-aware business intelligence [32]. Saggion and colleagues [34] employ ontologies for information extraction for business intelligence. Combining reasoning over ontologies and OLAP, Nebot and colleagues [24,25] build data warehouses for the analysis of semantic web data. The potential of ontology-based querying in business intelligence has been identified by Spahn and colleagues [40] but has not been elaborated for OLAP and multi-dimensional data warehouses.

The remainder of this tutorial is organized as follows. In Sect. 2, we present the semCockpit architecture and introduce a simplified real-world use case. In Sect. 3, we present the fundamentals of an MDO, including semantic dimensions. In Sect. 4, we investigate the definition of measures and scores based on the concepts of the MDO. In Sect. 5, we describe how the MDO can be beneficially applied for ontology-based comparative OLAP, thereby extending the well-known OLAP operations dice, slice, drill-down and roll-up for comparative analysis and making use of the MDO. In Sect. 6, we present a vision of BI analysis graphs for the definition of analysis processes. In Sect. 7, we introduce corresponding judgement and analysis rules for representing knowledge of the analyst.

[1] http://www.tableausoftware.com
[2] http://www.oracle.com

2 The semCockpit Approach

In this section, we describe the data warehouse behind a comparative data analysis project, introduce a case study, and identify the steps in a comparative data analysis project.

2.1 Data Warehouse

A data warehouse (DWH) typically organizes data as multi-dimensional facts, where each fact represents a business event that has been recorded in a transactional database, is identified by a node for each dimension, and described by one or several measures. Each dimension is given by a leveled hierarchy of nodes, whereby each node is described by a set of non-dimensional attributes and all nodes of the same level have the same attributes with different attribute values. The "base facts" of a data warehouse refer in each dimension to a leaf node of the dimension.

More specifically, a *data warehouse* consists of a set of dimensions and a set of fact classes. Each *dimension* has a dimension schema and a dimension instance. A *dimension schema* consists of a set of levels and a set of attributes for each level. Roll-up relationships organize the levels in a lattice. A *dimension instance* consists of a set of nodes where each node belongs to exactly one level and is described by a value for each attribute of the level. The nodes of a dimension are organized in a roll-up relationship that forms a semi-lattice such that each node of some level rolls up to exactly one node of each level that is in roll-up relationship to the level of the former node. Each dimension contains a single *top* level with a single *all* node to which all levels and all nodes of the dimension roll-up to. A *base fact class* consists of a fact schema and a set of facts. A *base fact schema* is given by a set of dimension roles and a set of measures (referred to as base measures). Each dimension role refers to a dimension schema (whereby two different dimension roles can refer to the same dimension schema). A *base fact* of a fact class refers for each dimension role to a leaf node in the respective dimension and is described by a measure value for each measure of the fact schema.

Example 1 (Existing Data Warehouse). Figure 1 illustrates a fragment of a simplified data warehouse schema of Austrian public health insurers, represented in a slight variation (explained later) of the Dimensional Fact Model (DFM) [12]. The dimensions are Insurant, Doctor, DrugATC (drug classification in accordance with ATC), and Time. *Top* levels of dimensions are not depicted. Medical sections (level medSec in dimension Doctor) denote specialisms of doctors, e.g., general practitioner (GP), internist, oculist. ATC is an abbreviation for Anatomical Therapeutic Chemical Classification System, which is an international classification system of drugs with the hierarchy levels for anatomical main group (atcAnatom), therapeutic main group (atcTherap), therapeutic/pharmacological subgroup (atcPharm), chemical/therapeutic/pharmacological subgroup (atcChemSubGr), and chemical substance. We omit the level for chemical substance.

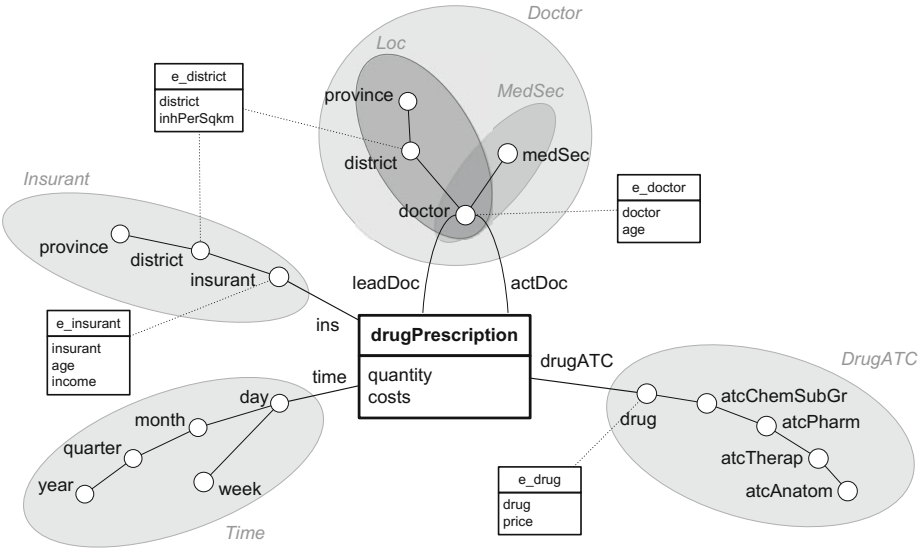

Fig. 1. DWH schema for drug prescriptions

The dimension roles of fact class drugPrescription are ins (insurant), leadDoc (lead doctor), actDoc (acting doctor), drugATC (drug classification according to ATC), and time. Dimension roles leadDoc and actDoc refer to the same dimension (Doctor). The fact class comprises measures quantity and costs of drug prescriptions.

In order to avoid redundant representations of attributes of levels the semDWH data model does not represent attributes with levels but with entity classes, where each level of a dimension schema refers to an entity class (and each node of that level to an entity of the entity class) and two levels of different dimensions (but not of the same dimension) can refer to the same entity class.

Example 2 (Entity Classes). Level district of dimension Doctor and level district of dimension Insurant refer to entity class e_district which specifies attributes district (the name of the district which is used as external identifier) and inhPerSqkm (the population density of the district).

As a means for specifying the domain of measures and multi-dimensional concepts, the semCockpit data model introduces the notion of a *dimension space*. A dimension space is defined by a set of dimension roles. A *point* in a dimension space is identified by a set of coordinates, one coordinate for each dimension role. Each coordinate of a point refers to a node of the dimension referred to by the dimension role. A *fact* is a point described by measure values. A dimension space comprises a point for each tuple in the cross product of the nodes in the dimension roles of the dimension space. A *granularity* in a dimension space is identified by a set of levels one for each dimension role. A lattice of granularities, from finer to coarser, can be derived from the hierarchies of levels of the dimensions referred to by the dimension roles. A dimension space may be restricted to

points at a particular granularity or to points that fall between a "from" granularity and a "to" granularity (both inclusive). Each fact class is defined over a dimension space restricted to a single granularity.

Example 3 (Dimension Space). In Fig. 1, the domain of measures quantity and costs of fact class drugPrescription is given by a dimension space which is defined by dimension roles time, ins, leadDoc, actDoc, and drugATC, restricted to the finest granularity [time:day, ins:insurant, leadDoc:doctor, actDoc:doctor, drugATC:drug]. A point in this dimension space, for example ⟨time:20130708, ins:mrHuber, leadDoc:drMaier, actDoc:drMueller, drugATC:paracetamol500mg⟩, may be described by base measures quantity and costs. Roll-up to coarser granularities together with aggregation of measures (for example SUM(costs)) may be possible to all points in the dimension space which is given by the same dimension roles but not restricted to the finest granularity, for example also to points like ⟨time:2013, ins:linz, leadDoc:GP, actDoc:all, drugATC:all⟩.

Each dimension is organized into one or more *roll-up hierarchies* (also referred to as *roll-up paths* or simply *hierarchies*). Per default, these hierarchies are alternative hierarchies, that is, each point in the dimension space is identified by exactly one coordinate per dimension role. The semDWH data model also allows for parallel roll-up, that is, points may have separate coordinates for the different roll-up paths of a dimension. Dimension spaces may then be defined over *dimension roles* and *hierarchy-specific dimension roles*. A hierarchy-specific dimension role is defined over a named roll-up path of a dimension.

Example 4 (Roll-up Hierarchies). In Fig. 1, dimension Time has two alternative hierarchies, one with a roll-up path along day, month, quarter, and year, and the other one along day and week, which cannot be used simultaneously in one query. Dimension Doctor has named roll-up paths Loc (location) and MedSec (medical section). Hierarchy-specific dimension roles actDocMedSec and actDocLoc as well as leadDocMedSec and leadDocLoc can be used to define dimension spaces that allow for parallel roll-up.

To simplify the later definition of concepts and their use in measure and score definitions and applications, we assume that dimension roles that are used in multiple fact classes carry the same meaning in each of these fact classes. This approach is akin to the unique role assumption for attributes in relational database systems. To support the unique role assumption for dimension roles, the semDWH data model provides for the definition of a *universal dimension space* consisting of all dimension roles of a semDWH, where each dimension role is identified by a unique name and described by the dimension it refers to. The universal dimension space also comprises all hierarchy-specific dimension roles. All other dimension spaces may not contain both, a dimension role and one of its hierarchy-specific dimension roles.

Example 5 (Universal Dimension Space). Fact class ambTreatment (ambulant treatments) has the same dimension roles as fact class drugPrescription but it

possesses dimension role medServItem (medical service items) instead of dimension role drugATC. Examples of medical service items are doctor visits and blood glucose examinations. The universal dimension space comprises the dimension roles of both fact classes, drugPrescription and ambTreatment. The dimension spaces DrugPrescriptionSpace and AmbTreatmentSpace only contain the dimension roles of the fact classes drugPrescription and ambTreatment, respectively.

In general, OLAP operations allow to join facts over different dimension spaces in drill-across operations. The result of such an operation is a fact over a new dimension space, whereby the dimension roles of the drill-across fact are mapped to the dimension roles of the joined facts. Accordingly, a *drill-across* dimension space can be defined based on two other dimension spaces (by mapping dimension roles based on common names like in natural joins for relations, or explicitly, like in equi-joins). It comprises the union of dimension roles.

The semCockpit data warehouse (semDWH) can be defined in two ways. First, the semDWH can be defined from scratch using a simple data definition language (DDL). Each DDL statement of a semDWH construct has a corresponding relational representation (for dimensions and fact classes). Actual data have to be provided thereafter as materialized or virtual views over the enterprise DWH according to this relational representation. We do not describe this schema and instance mapping problem and refer to the relevant literature [4] instead. Second, the semDWH can be defined by immediately generating the appropriate relational representation (i.e., materialized or virtual views) of the semDWH. The former approach is more appropriate for reuse in similar settings (e.g., health insurers in different states of Austria), the latter for single in-house projects.

2.2 Use Case

Effective and efficient medical care is an overall goal of public health insurance companies. Comparative data analysis often focuses on diseases that are responsible for high overall costs. For example, diabetes mellitus of type 2 (DM2) is one specific example of a lifestyle disease with high prevalence that causes high costs. Disease management programs (DMP) have been established to provide effective and cost efficient treatment of DM2 patients.

In this context the managerial accounting department of a public health insurer might recognize an above-average increase of total costs concerning the treatment of DM2 patients. The management asks the business intelligence department to analyze the issue by finding striking differences through comparison. In the subsequent comparative data analysis processes, a business analyst may compare different groups of patients (rural vs. urban districts, young vs. old, DMP patients vs. non-DMP patients, etc.), different groups of doctors, different drugs, different insurers, or different periods.

The analysis process is interactive, exploratory, and iterative. The analyst starts with a vague analysis question and interacts with various domain experts

to discuss what kinds of comparison might be relevant or should be chosen next. The interesting comparisons develop over time. Once successful and relevant sequences of analysis steps have been discovered, they can be described in generalized form for later re-use in analogous situations (e.g., for other years, insurers, or diseases).

2.3 Steps in a Comparative Data Analysis Project

Comparative data analysis focuses on the interactive comparison of various sets of data. Such comparisons are based on measures and scores. A measure describes a multi-dimensional point which consists of nodes from data warehouse dimensions; a point and a measure together give a *fact*. A score describes a relationship between a pair of points, the point of interest and the point of comparison; score, point of interest (PoI) and point of comparison (PoC) together give a *comparative fact* which explicitly expresses the result of a comparison that would otherwise have been left to the human eye. Thus, relating our DWH model to the Entity-Relationship model, points correspond to entities, measures to attributes of entities, and scores to attributes of binary relationships between entities. Since a comparative fact actually relates, via aggregation, sets of facts that are compared, we speak also of group of interest (GoI) and group of comparison (GoC).

The typical steps in applying an ontology-driven business intelligence approach for comparative data analysis are: (1) *Model transformation*, in which a given DWH schema is transformed into a semDWH data warehouse schema as introduced in Subsect. 2.1, (2) *Semantic Enrichment*, in which business terms are expressed as concepts in a multi-dimensional ontology (MDO) or imported from an external domain ontology, (3) *Calculation Definition*, in which measures and scores are defined by calculations over other measures and scores, employing concepts of the MDO to select the data to be included in calculations, (4) *Explorative Measure and Score Application*, in which measures and scores are applied to different points of interest and comparison, (5) *Analysis Design*, in which promising sequences of measure and score applications are modeled as BI analysis graphs for later re-use, (6) *Rule Design*, in which different kinds of rules are designed to complement analysis: (a) guidance rules that provide context-sensitive semantic guidance on which path to follow in an instantiation of an analysis graph, (b) judgement rules that provide background information about possible reasons for a striking score, and (c) analysis rules that express how to react on specific measures and scores detected (action rules) or provide a concise analysis report (reporting rules), (7) *Proper comparative data analysis*, in which BI analysis graphs are instantiated and traversed for particular analysis problems, thereby possibly backtracking to any of the previous steps. – Steps 1 to 4 and 6 (b,c) have been investigated in the semCockpit project and implemented in our semCockpit prototype. Steps 5, 6 (a), and 7 have been identified during the project as beneficial future extensions to capture and exploit – next to "static" knowledge – knowledge about analysis processes.

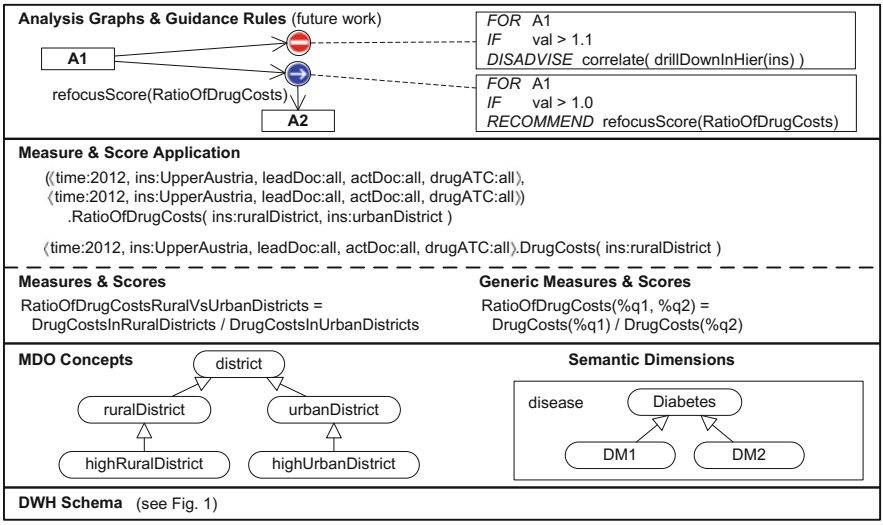

Fig. 2. The semCockpit stack

The semCockpit stack (Fig. 2) reflects the typical steps in the application of an ontology-driven BI approach for comparative data analysis. Its starting point (bottom of Fig. 2) is founded on the DWH schema as described in example 1.

MDO concepts are defined by logical expressions over nodes of a dimension. MDO concepts are organised in subsumption hierarchies through reasoning over concept expressions. Furthermore terms of an external domain ontology such as SNOMED CT[3] can be used as semantic dimensions in the way that leaf concepts of the external ontology classify facts.

Example 6 (MDO concepts and semantic dimensions). The MDO concepts ruralDistrict and urbanDistrict in Fig. 2 are defined over nodes of a location dimension (logical definitions are omitted), organized in a subsumption hierarchy. The semantic dimension disease incorporates an OWL representation of a subset of SNOMED CT.

Measures describe by measurement instructions how a particular measure value is calculated from facts in the DHW for a point. Measures are *ontology-based* in the sense that measurement instructions refer to MDO-concepts to identify facts to be included in (parts of) calculations. Similarly, scoring instructions describe for scores how a score value is calculated for a pair of points.

Example 7 (Measures and Scores). Figure 2 shows two measures, DrugCostsInRuralDistricts and DrugCostsInUrbanDistricts, that calculate the total costs for drugs in rural and urban districts, respectively. The measurement instructions (omitted) refer to the respective ontology concepts, ruralDistrict and urbanDistrict.

[3] Systematized Nomenclature Of Medicine Clinical Terms.

The score RatioOfDrugCostsRuralVsUrbanDistricts compares the total costs for drugs in rural districts against urban districts (by using the appropriate measures in the scoring instructions, not shown).

Generic measures and scores avoid the need of repeated definitions of instructions that are identical apart from a particular MDO-concept. They are defined with parameters for concepts. We refer to these parameters also as (generic) qualifiers as they are used to select facts.

Example 8 (Generic Measures and Scores). Generic measure DrugCosts has a qualifier, which can be instantiated for example by a location concept. If instantiated, e.g., with ins:urbanDistrict, the instantiation gives a non-generic measure, e.g., DrugCostsInUrbanDistricts. Similarly, generic score RatioOfDrugCosts has two qualifiers, one for the group of interest, one for the group of comparison.

Once defined, measure and scores can be applied to multi-dimensional points (shown in the next stage of the semCockpit stack). Additionally, they can be qualified by MDO concepts.

Example 9 (Measure and Score Application). Figure 2 shows an application of generic measure DrugCosts to multi-dimensional point ⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩ with actual qualifier ins:ruralDistrict giving the total drug costs prescribed in rural districts in Upper Austria in the year 2012. The application of generic score RatioOfDrugCosts to group of interest ⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩ with qualifier ins:ruralDistrict and group of comparison ⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drug ATC:all⟩ with qualifier ins:urbanDistrict returns the ratio of drug costs of Upper Austrian patients of rural versus urban districts in year 2012.

An analysis graph has analysis situations as nodes and analysis steps as arcs and describes promising analysis processes. Analysis situations are parameterized cubes, analysis steps are parameterized navigation operations, e.g., drill-down. The instantiation of the parameters leads to a specific BI analysis.

Depending on measure and score values, guidance rules open or close analysis paths. Judgement rules provide background information on striking score values, and analysis rules describe how to react upon (action rules) or which specific comparative facts to report (reporting rules) from a set of comparative facts provided (e.g., those loaded in the last ETL cycles).

Example 10 (Analysis Graphs and Rules). The top compartment of Fig. 2 depicts a simple analysis graph consisting of two analysis situations, A1 and A2, and two guidance rules. For example, A1 may represent the ratio of ambulant treatment costs for some selected comparison. If the result is greater than 1.1, the associated guidance rule disadvises to drill down in the insurant hierarchy. If the result is greater than 1.0, the associated guidance rule suggests to refocus in A2 the analysis on the ratio of drug costs. The figure assumes that for the currently selected comparison A1 the first rule does not apply and the second rule applies.

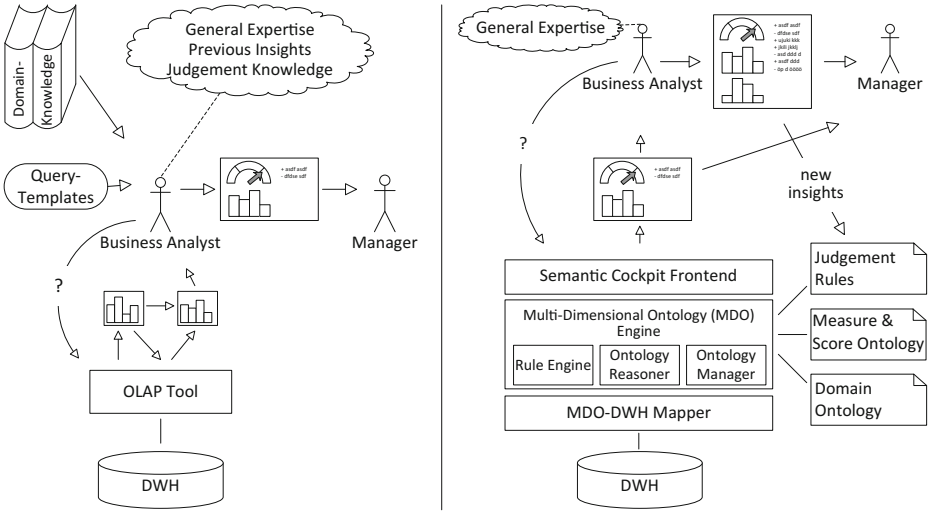

Fig. 3. Conventional Comparative Data Analysis (left) vs. Semantic Cockpit: Setting & Components (right)

Figure 3 gives an overview of the semCockpit architecture. The left-hand side illustrates conventional comparative data analysis in which a business analyst uses OLAP tools on an operative level and domain knowledge is not represented. The right-hand side illustrates the semCockpit approach in which domain knowledge is captured by a multi-dimensional ontology (MDO) that describes relevant business terms in the context of business analysis and which may relate to a domain ontology. Measures and scores are defined based on the ontology. Previous insights and analysis experience are captured by judgement rules. In order to implement these features, semCockpit comprises several components. The MDO-DWH Mapper accesses DWH data that is used by other components. The MDO Engine administers concepts, measures and scores, and organizes them by reasoning. The management and evaluation of judgement rules is carried out by the Rule Engine. Finally the semCockpit Frontend provides an appropriate user interface. – The planned future extensions, analysis graphs and guidance rules, are not shown.

3 Multi-Dimensional Ontology

In this section, we present the representation of business terms as concepts of a multi-dimensional ontology (MDO), the handling of context-specific concepts, the seamless import of a domain ontology into the MDO as *semantic dimension*, and the translation of concepts into SQL for querying and into OWL for determining subsumption hierarchies by OWL reasoners.

In the semCockpit approach, a multi-dimensional ontology is managed as a central repository of business terms that are defined and maintained collaboratively by business analysts. Business terms can be translated automatically

to SQL and simplify the formulation of otherwise complex multi-dimensional queries. Automatically-derived subsumption hierarchies simplify the organization of business terms and the detection of similar or redundant concepts. In analysis sessions, subsumption hierarchies allow to move up or down the hierarchy to 'broaden' or to 'narrow' the selection predicate of a query.

With regard to querying, the semCockpit approach assumes that the data in the underlying data warehouse is complete (closed world assumption). With regard to subsumption reasoning, the approach does not consider the complete data in the data warehouse but only the incomplete knowledge represented in the ontology (open world assumption) because subsumption hierarchies should not change due to changes in the data of the data warehouse.

Using ontologies with defined classes for querying data- or knowledge bases has seen considerable attention in the literature. Staudt et al. [41] discuss the definition of query classes in deductive databases. The partitioning of the terminological box of an ontology into a schema part and a view part, with distinct language constructs for either part, as proposed by Buchheit et. al. [5], is of particular importance for MDOs. The MASTRO system [6] for ontology-based data access uses ontologies for querying incomplete (relational) databases. Lim et al. [22] employ virtual views as a query interface for semantically-enriched relational data.

3.1 Concepts: Signatures and Concept Expressions

The MDO enriches the underlying semDWH by a set of concepts representing business terms and their meaning in the context of data analysis. A concept may be defined over (a) entities of an entity class (*entity concepts*), (b) nodes of a dimension (*dimensional concept*), (c) points of a dimension space (*multi-dimensional concept*, *md-concept*), or (d) pairs of points, referred to as point of interest (PoI) and point of comparison (PoC), (*comparative concept*).

Each concept has a *signature* and a *membership condition*, which may be defined independently of each other. The separation of signature and membership condition is a prerequisite for specializing membership conditions for different contexts (see next subsection).

The *signature* of a concept is given by a name and an interpretation domain for the concept. The interpretation domain is the set of individuals for which the concept is defined. It is given by the sort of individuals over which the concept is defined (entities of an entity class, nodes of a dimension, points of a dimension space, point-pairs of a comparative space consisting of two dimension spaces) and is possibly restricted to some subset of the sort. E.g., the interpretation domain of a dimensional concept may be restricted to nodes of some level or nodes that fall into a level range. A point satisfies a multi-dimensional concept, if the point satisfies the concept for the dimension roles for which it is defined. (Notice that this interpretation is consistent with classifying individuals in ontologies where properties not referred to in a concept expression are ignored.) This holds analogously for pairs of points and comparative concepts. The signature of a

concept does not need be explicitly stated but may also be derived from its membership condition, which is a concept expression.

Example 11 (Signature). Signature ruralDistrict(e_district) describes an entity concept called ruralDistrict with interpretation domain entity class e_district. InOAD(DrugATC[drug .. atcPharm]) represents the signature of a dimensional concept. It indicates that concept InOAD refers to dimension DrugATC and comprises nodes between level drug and atcPharm (both inclusive). Signature PatInRuralDistrLeadDocInUrbanDistr(ins[insurant .. district], leadDoc[doctor .. district]) represents the multi-dimensional concept PatInRuralDistrLeadDocInUrbanDistr that comprises points over insurants (dimension role ins) and lead doctors (dimension role leadDoc) at granularity range from level insurant to district and from level doctor to district, respectively. The signature PatWithRegularDocVisitsInYear(ins[insurant], time[year]) restricts the multi-dimensional concept PatWithRegularDocVisitsInYear to level insurant for dimension role ins and to level year for dimension role time.

The membership condition (also often referred to as necessary & sufficient condition) is given by a concept expression in a simple, high-level MDO language (surveyed below) or by an SQL view over the underlying semDWH. SQL-defined concepts provide for extensibility and are treated as primitive when determining subsumption hierarchies between concepts (see Subsect. 3.4). The MDO language has been designed to support the most important use cases and to provide for a mapping [28] into OWL 2 DL.

Example 12 (Entity concepts on district). The concepts ruralDistrict, urbanDistrict, highRuralDistrict, and highUrbanDistrict presented in Fig. 4 are defined over entity e_district. Membership conditions are denoted next to the box of the concept, e.g., inhPerSqkm <= 400 represents the expression for concept ruralDistrict. As discussed later, the reasoner will detect that concept highUrbanDistrict is subsumed by concept urbanDistrict because expression inhPerSqkm $>$ 1000 implies inhPerSqkm $>$ 400. In Fig. 4 inferred subsumption relationships between concepts are denoted by arrows. Dotted lines link entity concepts to their entity class. For better readability this type of lines are omitted for subsumed concepts like highUrbanDistrict.

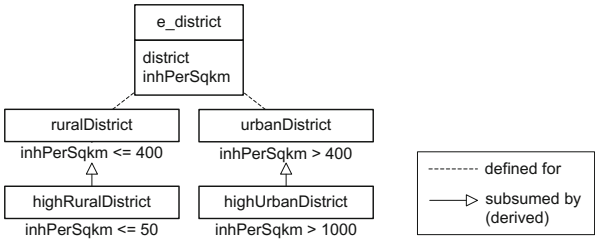

Fig. 4. Entity concepts on district

The membership condition of a dimensional concept is given by one of the following kinds of concepts expressions: (1) by a reference to an entity concept (such that each node that refers to an entity satisfying the entity concept is in the interpretation of the defined concept), (2) by hierarchy expansion of some concept (such that each node of the dimension that is in the interpretation of the concept or some direct or indirect successor node thereof is in the interpretation), (3) by level range restriction of some concept (such that only nodes of that concept that fall in between an indicated top and bottom level, inclusive, are in the interpretation), (4) by intersection, union, or complement (open world interpretation) of concepts defined for the same level (with the usual interpretation), (5) as ⟨node⟩concept[level-or-levelRange]-expression (such that all nodes that satisfy the hierarchical expansion of the indicated concept, are at the indicated level(s), and beneath the indicated node are in the interpretation). Notice that the construct ⟨node⟩concept[level-or-levelRange]-expression does not enhance the expressiveness of MDO, but assists in structuring concept expressions in an OLAP setting along modeling elements of DWH dimensions: hierarchy of nodes, properties of nodes (via entities), and levels.

Example 13 (Dimensional concepts). Figure 5 shows concepts for DM2 specific drugs: InAD (to be read as "in antidiabetic drug group"), InOAD (to be read as "in oral antidiabetic drug group"), InStarterOAD (comprises OAD drugs that should be used first when diagnosis DM2 is determined). The concept expression of InOAD denotes that at level atcPharm ATC code A10B is selected and the asterisk on the right of the expression denotes hierarchical expansion, i.e., all subnodes below node A10B belong to concept InOAD. On the left hand side of Fig. 5 one can see the level hierarchy and on the right hand side a selection of the node hierarchy of the dimension. Items bordered by continuous lines (InAD, InOAD, and InStarterOAD) represent hierarchical concepts, which comprise nodes of multiple

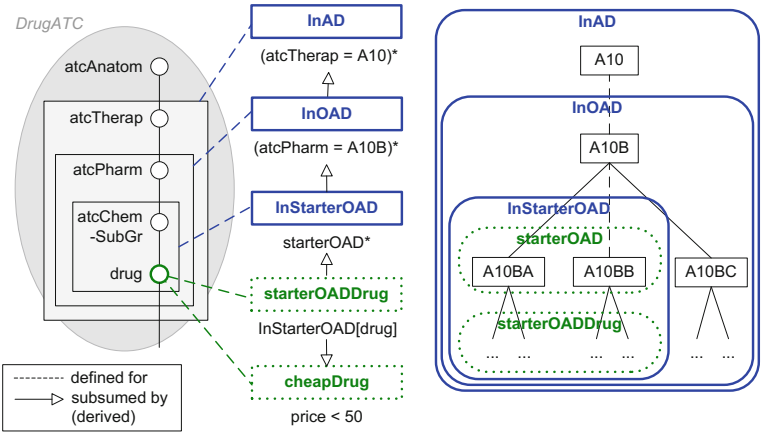

Fig. 5. Dimensional concepts over drugs

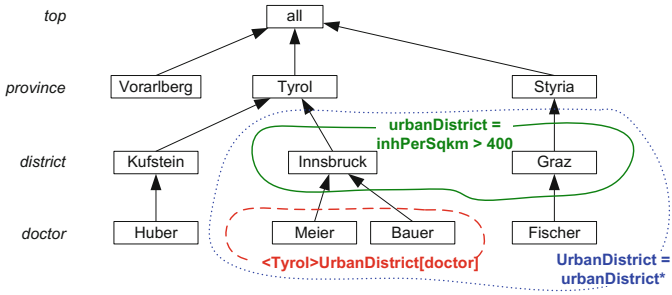

Fig. 6. ⟨point⟩concept[granularity]-expression

levels. Items bordered by dotted lines (starterOAD, starterOADDrug, and cheapOAD) denote flat concepts, which only comprise nodes of one level. Additionally derived subsumption relations are shown, e.g., cheap drugs subsume starter OAD drugs. Figure 6 illustrates the interpretation of a ⟨node⟩concept[level]-expression. Concept ⟨Tyrol⟩UrbanDistrict[doctor] comprises nodes of dimension Doctor at level doctor who live in an urban district in province Tyrol. Concept UrbanDistrict is the hierarchical expansion of concept urbanDistrict, comprising districts with more than 400 inhabitants per square km.

The membership condition of a multi-dimensional concept (md-concept) is given in one of the following ways: (1) by reference to a dimensional concept for some dimension role (in which case all points that satisfy the dimensional concept in the indicated dimension role are in the interpretation), (2) by hierarchy expansion of a md-concept (such that each point that is in the interpretation of the md-concept or a descendent thereof is in the interpretation), (3) by granularity restriction of some md-concept (such that only points of the indicated md-concept that are between a given top and bottom granularity are in the interpretation), (4) by intersection, union, or complement (open world interpretation) of md-concepts defined over the same dimension roles and the same granularity (usual interpretation), (5) as ⟨point⟩concept[granularity-or-granularityRange]-expression (such that all points that satisfy the hierarchial expansion of the indicated concept, are at the indicated granularity, and beneath the indicated point are in the interpretation), or (6) by a boolean expression over measure-value comparisons of measures applied to a point (*fact-based concept*). Further, each dimension space is also a md-concept.

Example 14 (Multi-dimensional concepts). The concept expression of multi-dimensional concept InsInRuralDistrLeadDocInUrbanDistr indicates that points that refer in the dimension role ins to urban districts and in the dimension role leadDoc to urban districts are interpretation as well as points that roll up to such points (Fig. 7). The expression uses concepts ins:ruralDistrict and leadDoc:urbanDistrict. Multi-dimensional concept InsInHighRuralDistrLeadDocInUrbanDistr is defined in a similar way. As discussed later, the subsumption relationship between both concepts can be detected by reasoning.

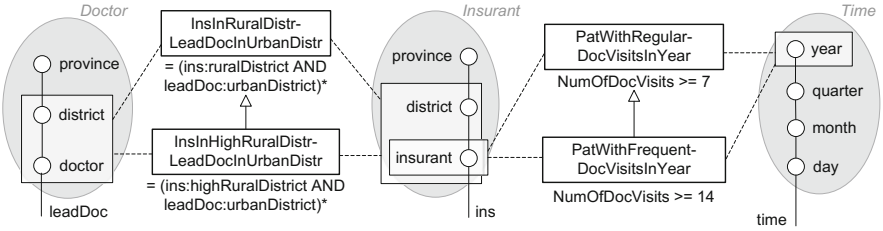

Fig. 7. Multi-dimensional concepts

Example 15 (Fact-based multi-dimensional concepts). PatWithRegularDocVisitsInYear is a fact-based multi-dimensional concept (Fig. 7). It comprises points of patients and years, for (patient, year)-pairs such that the patient had at least seven doctor visits in the year. The restriction to levels insurant and year is necessary because the concept as such is meaningful only for a single patient and a given year. The second fact-based multi-dimensional concept PatWithFrequentDocVisitsInYear comprises patients having at least fourteen doctor visits a year, a corresponding subsumption relationship will be inferred (as mentioned already before). The concept expressions assume that an aggregate measure NumOfDocVisits (number of doctor visits) over patients and years has been defined (Measure definition is described later).

The membership condition of a comparative concept is given like multi-dimensional concepts, with the addition that it may be also given by (a) two md-concepts, (b) a join condition relating nodes of the PoI and the PoC by a conjunction of pre-defined comparison predicates such as equality or predecessor/successor-relationships, and (c) by a boolean expression over score-value comparisons of scores applied to PoI and PoC.

3.2 Context-Specific and Contextualized Concepts

We present now a kind of specialization for concepts that is akin to specialization in object-oriented systems employing the abstract superclass rule [17].

In object orientation, the signature or head of a method may be introduced in an abstract superclass without providing an implementation of the method. The implementation (also referred to as body) of the method is provided by each concrete subclass of the abstract superclass. In an MDO, in analogy to object-orientation, a concept may be regarded as a boolean method of its domain (a concept may be applied on each node or point in its domain and returns either true or false) where the domain of the concept is analogous to the class in which the method is introduced and where the membership condition of the concept is analogous to the implementation of the boolean method. A contextualized concept is analogous to a boolean method introduced at some abstract superclass (with the domain of the contextualized concept playing the role of the abstract

superclass). Context-specific concepts are analogous to boolean methods at subclasses that implement the method introduced at the abstract superclass, with the context of a context-specific concept, which is given by an MDO concept expression, being analogous to the subclass at which the boolean method is implemented.

Context-specific concepts are defined over a selected subset of nodes of a dimension (dimensional concepts) or over a subset of points of a dimension space (md-concepts) by indicating a *context* in the signature. Thereby a *context* is given by ⟨node⟩concept[level-or-levelRange]-expression (for dimensional concepts) or a ⟨point⟩concept[granularity-or-GranularityRange]-expression (for md-concepts).

Contextualized concepts are defined by several context-specific concepts. The contexts of these concepts must cover all points in the signature of the contextualized concepts such that each point belongs to exactly one most specific context (i.e., the point does not belong also to another context subsumed by the former).

Example 16 (Contextualized concepts). Different to concept PatWithRegularDocVisitsInYear in example 15, we now consider regular visits of a patient to a particular doctor in a year. Regularity of visits depends on the medical section of a doctor. We assume, for simplicity, that there are only two medical sections, namely general practitioner and oculists. Suppose a regular patient of a general practitioner (GP) must have at least four GP visits per year whereas for a regular patient of an oculist it is sufficient to have at least two visit per year. Of course, one could define two separate concepts, but we specify one contextualized concept PatWithRegularVisitsToDocInYear([time:year,ins:insurant,actDoc:doctor]) consisting of two context-specific concept definitions: NumOfDocVisits ≥ 4 for context ⟨time:all, ins:all, actDoc:GP⟩[time:year, ins:insurant, actDoc:doctor] and NumOfDocVisits ≥ 2 for context ⟨time:all, ins:all, actDoc:oculist⟩[time:year, ins:insurant, actDoc:doctor]

3.3 Semantic Dimensions

Transaction systems collect records that refer to concepts of domain ontologies in semantic attributes. For example, medical treatment records may refer to a diagnosis code of SNOMED CT. In order to exploit semantic attributes for OLAP-style analysis, we wish to use the existing domain ontology to which semantic attributes refer like a common dimension in data warehousing and call a dimension based on a domain ontology in analogy to semantic attributes, a semantic dimension. A semantic dimension will be usually expressed or mapped to an ontology language such as OWL.

Querying over semantic attributes in relational databases is discussed by Das et al. [8] and implemented in *Oracle Database 11g Semantic Technologies*. In addition to their approach, the semCockpit approach also allows to use concept expressions as selection criteria (post-coordination) and provide for a seamless integration in data warehousing and OLAP. The challenges tackled by this approach are akin to the challenges of heterogeneous dimensions [18,21,27].

Malinowski and Zimányi [23] give an overview of different kinds of dimension hierarchies.

We briefly explain how domain ontologies can be used as semantic dimensions, and refer to [1] for a more elaborate treatment: (1) The existing concepts (usually called pre-coordinated concepts) are mapped to nodes of a dimension hierarchy. (2) Facts may refer to leaf or inner nodes. The meaning of a fact refering to an inner node "c" is "c only", e.g., "DM-2 without further information on the subkind of DM-2". The latter concept ("c only") is not represented in the original domain ontology, but would be a leaf node and a child of the former ("c"). (3) New concepts (usually called post-coordinated concepts) may be defined upon existing ones in the external domain ontology language (e.g., by OWL expressions) and are mapped to an MDO concept. An MDO concept C corresponding to concept c in the external domain ontology comprises all nodes that correspond to a concept subsumed by c. (4) While the domain ontology is un-leveled, levels may be introduced explicitly by identifying the concepts (nodes) that make up the members of a level. To provide for summarizability (i.e., ensuring that the sum over all base facts of an additive measure is the same than the sum of the aggregated measure over all roll-up facts at some granularity), the member concepts (nodes) of such a level must be disjoint (i.e., have non-overlapping interpretations) and be complete with respect to the bottom level (i.e., each leaf node must be a member of some member concept). (5) Levels may be defined context-specific, i.e., local to a node. (6) Built that way, precoordinated concepts can be used like native nodes and post-coordinated concepts like MDO concepts over native dimensions for OLAP operations slice, dice, and roll-up.

The treatment of native and semantic dimension becomes seamless by unifying native dimensions and semantic dimensions. A node of a dimension may relate to either an entity (entity node) or to an entity concept (concept node), which may be given by an entity concept as introduced above or by a domain ontology concept. Levels consist either of entity nodes (entity levels) or concept nodes (concept levels), which may be introduced above an entity level.

Example 17 (Semantic Dimension – SNOMED CT). Figure 8 shows a small part of the SNOMED CT hierarchy. It can be taken to implement a semantic dimension for disease which can be linked to fact classes (e.g., drugPrescription and ambTreatment). In Fig. 8 we have already extended the SNOMED CT concepts with "only"-concepts, i.e., each inner node has as an "only"-node as a subconcept. E.g., the node Diabetes mellitus has the additional subnode Diabetes mellitus only, which does not exist in the original SNOMED CT hierarchy. A fact can refer all leaf nodes presented in our diagram, i.e., all original leaf nodes and all "only"-leaf nodes (original inner nodes). The figure also illustrates how conventional OLAP operations can be applied for semantic dimensions. The nodes bordered by the continuous line represent the result of a DICE operation that selects the subhierarchy under node Diabetes mellitus (analogously to conventional OLAP operation DICE which selects a subcube). Nodes bordered by the coarse dotted line represent the result of a SLICE operation with condition "all nodes

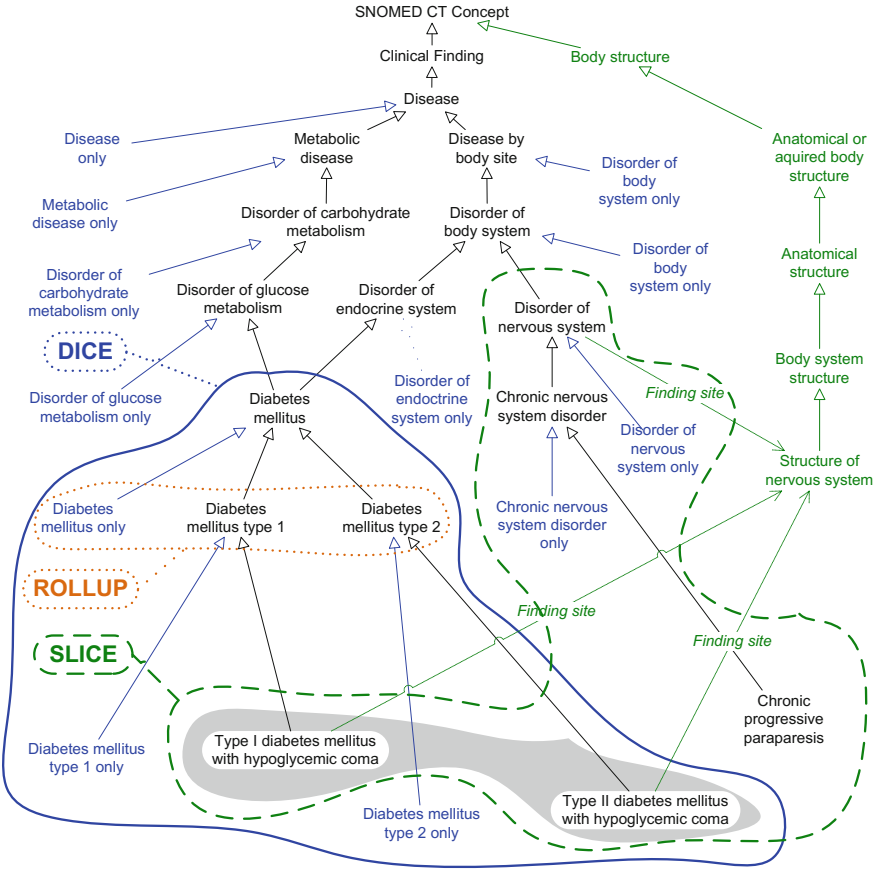

Fig. 8. Semantic Dimension (SNOMED CT)

for which there exists finding site equal to Structure of nervous system". Applied to DICE node Diabetes mellitus the SLICE operation returns the grey filled area that comprises the nodes Type I diabetes mellitus with hypoglycemic coma and Type II diabetes mellitus with hypoglycemic coma. Finally one can ROLLUP to nodes Diabetes mellitus only, Diabetes mellitus type 1, and Diabetes mellitus type 2 (bordered by the fine dotted line) that represent a virtual level local to Diabetes mellitus.

3.4 Relational and OWL Representations

MDO concepts are translated to SQL for querying the underlying closed world data warehouse. MDO concepts are translated to OWL in order to delegate subsumption checking to an off-the-shelf OWL reasoner. The mappings for each kind of concept (apart from comparative concepts) are given in [28].

The relational representation of MDO concepts builds on the relational representation of the semCockpit data warehouse where entity classes, dimensions

and dimension spaces are directly available through SQL tables and views. Entity concepts, dimensional concepts, and multi-dimensional concepts are translated to views over this relational representations of entity classes, dimensions, and dimension spaces as well as over views generated for previously defined concepts.

We now shortly explain the rationale of the representation of MDO concepts in OWL and refer to [28] for a more elaborate treatment. We assume a basic familiarity with OWL [16]. Using the OWL representation, the initially unordered set of business terms represented by the MDO may be automatically organized in subsumption hierarchies.

Example 18 (Concept organization). Figure 9 illustrates for a selection of our use case, how business terms are organized in subsumption hierarchies along the dimensions of a data warehouse.

Individuals of the ontology are MDO entities, nodes, levels, points, and point-pairs. MDO entities are collected into disjoint OWL classes, one OWL class for each MDO entity class. Attributes of MDO entities are represented in OWL as object properties of MDO entities. Nodes are collected into disjoint OWL classes, one OWL class for each dimension. Each node is associated with one level and with one MDO entity, this is represented by object properties atLevel and roleOf, respectively. Roll-up relationships between nodes as well as between levels are represented as transitive and reflexive property rollsUpTo. Dimension roles are represented as object properties of points, where the range of a dimension role is a dimension. Dimension roles of point-pairs are distinguished into PoI- and PoC-dimension roles. Also, an OWL class is defined for each dimension space and each comparison space. Notice that MDO entities and nodes not explicitly referred to in MDO concept expressions as well as points and point-pairs are not represented as named individuals in the OWL ontology.

MDO concepts are translated to OWL classes according to the MDO concept expression. To capture level-restrictions the level range is checked by a property restriction on property atLevel indicating that node's levels must drill-down to the "from-level" and roll-up to the "to-level".

One of the challenges of representing MDO concepts in OWL is to cope with the following restriction of OWL 2 DL[4]: It is not allowed to express that a transitive relationship such as rollsUpTo maps to exactly one object in a given range (e.g., to one node of one level). But, the essential characteristics of roll-up hierarchies of data warehouse dimensions are that (i) the rollsUpTo-relationship between nodes is transitive, and (ii) each node of a level rolls up to exactly one node of a higher level (to which the former level rolls up). Without this semantics of roll-up hierarchies in data warehousing being captured, OWL reasoners will not be able to recognize that certain concepts are disjoint. To cope with this limitation of OWL, for each level X there is a functional object property rollsUpTo_X. For each named node nd at level X there is a subclass axiom stating

[4] http://www.w3.org/TR/owl2-syntax/#Global_Restrictions_on_Axioms_in_OWL_2_DL

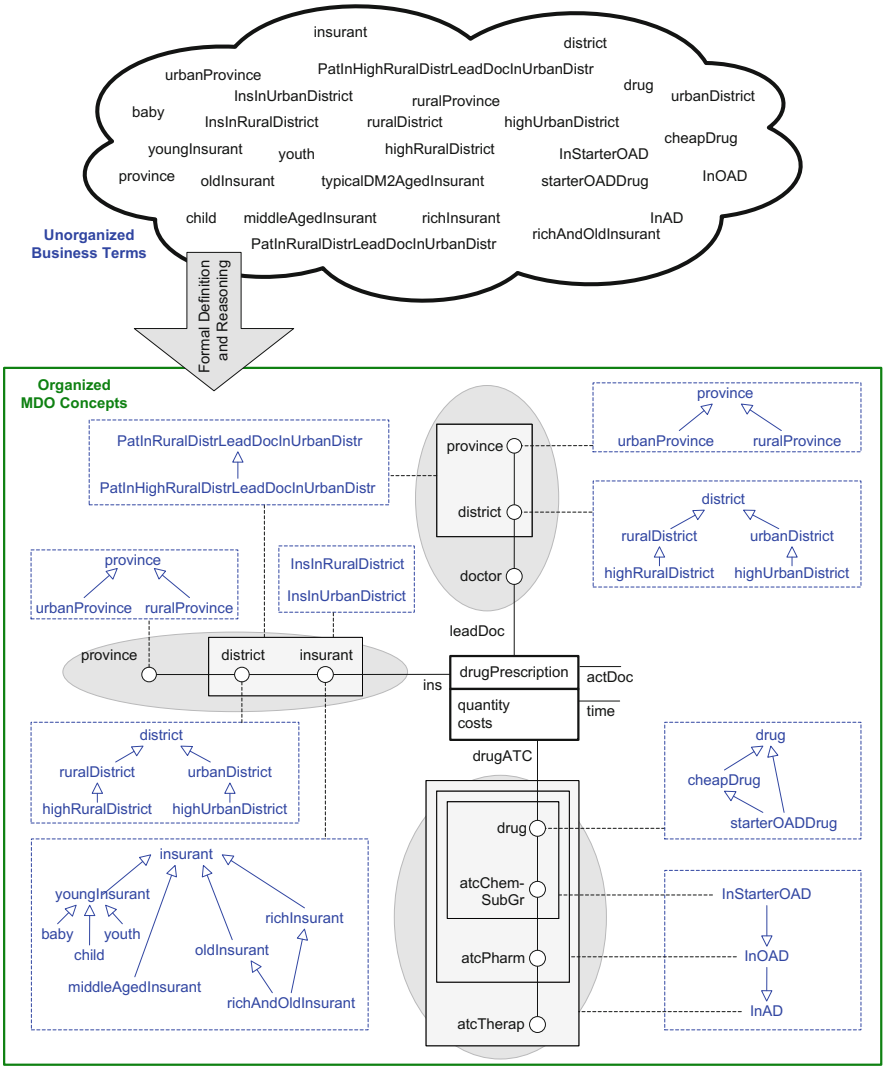

Fig. 9. Concept organization

that every descendant node of *nd* rolls up to *nd* via functional object property rollsUpTo_X.

Example 19 (Translation of MDO concepts to OWL). Multi-dimensional concept InsInRuralDistrLeadDocDoctorInUrbanDistr[ins: insurant..district, leadDoc: doctor] is interpreted by the set of points that each refer via dimension role ins to a node that rolls up to a node which is a role of a rural district and refer via dimension role leadDoc to a node at level doctor that rolls up to a node which is

a role of an urban district. In OWL this is represented (using Description Logics notation) as:

InsInRuralDistrLeadDocDoctorInUrbanDistr ≡
  ∃ins.∃rollsUpTo_district.∃roleOf.ruralDistrict ⊓
  ∃leadDoc.(∃atLevel.{doctor} ⊓∃rollsUpTo_district.∃roleOf.urbanDistrict)

4 Ontology-Based Measures and Scores

A *measure* is defined for points in a dimension space, which is also called the domain of the measure. Measures are distinguished into base and derived. Base measures are given as primitive and relate to a measure of a fact class (Note: To provide for parallel analysis across multiple roll-up hierarchies of a dimension role, several base measures may be defined for a measure of a fact class, each of them defined for a different dimensions space that replaces selected dimension roles of the dimension space of the fact class by one or several hierarchy-specific dimension roles). Derived measures have measurement instructions that describes how a measure value is calculated for each point in the measure domain from other measures.

A *score* is defined for pairs of points (PoI, PoC) in a comparison space. The scoring instruction of a score describes for each pair of points (PoI, PoC) in the score domain how a score value is calculated from measures.

A measure may be *applied* to a point for which it is defined, returning the measure value; a measure applied to a set of points gives a set of measure values. A measure may be applied to a point with more dimension roles for which the measure is defined; in such a case the superfluous dimension roles are ignored. We denote measure application by the '.'-operator.

Example 20 (Measure application). The following measure application returns the drug costs of patients in Upper Austria in the year 2012:

⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩.DrugCosts

Alike measure applications a score may be *applied* to a pair of points for which it is defined, returning the score value; a score applied to a set of pairs of points gives a set of scores. If a score is applied to a pair of points with more dimension roles as for which the score is defined, the superfluous dimension roles are ignored. As for measures we denote score application by the '.'-operator.

Example 21 (Score application). The following score application returns the ratio of drug costs of patients in Upper Austria in year 2012 (as point of interest) to the drug costs of patients in Upper Austria in 2011 (as point of comparison):

(⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩,
 ⟨time:2011, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩)
      .RatioOfDrugCosts

A *fact* is a point of a dimension space together with values for one or several measures; a *comparative fact* is a pair of points together with values for one or several scores.

A *cube* (*comparative cube*) is a set of facts (comparative facts), possibly at different granularities. Special kinds of cubes are (a) fact classes, which are mono-granular and primitive (i.e., its facts are not calculated from other facts in the DWH), (b) measure cubes and score cubes, which are cubes over the domain of a measure or score and whose facts possess only this measure or score, and (c) cuboids, which are mono-granular slices of another cube. Cubes - other than fact classes and measure cubes - are defined by a *cube space*, given by a md-concept, preferably in the form ⟨point⟩concept[granularity-or-granularity-range], and a set of measures that may be applied to points in the cube space to construct facts. This is indicated by applying a measure with the ".."-operator to the cube space and optionally indicating after the measure by the "\"-operator whether a null-value of the measure should be replaced by some other value.

Example 22 (Cube). A cube consisting of drug-costs facts for points at granularity [time:month, ins:district, leadDoc:district, actDoc:top, drugATC:top] that satisfy concept InsInRuralDistrLeadDocInUrbanDistr and roll-up to point ⟨time:2012, ins:Austria, leadDoc:Austria, actDoc:all, drugATC:all⟩, is defined by:

⟨time:2012, ins:Austria, leadDoc:Austria, actDoc:all, drugATC:all⟩
InsInRuralDistrLeadDocInUrbanDistr
[time:month, ins:district, leadDoc:district, actDoc:top, drugATC:top]
..DrugCosts

Likewise, a comparative cube is defined for a comparative cube space, given by a comparative concept, and a set of scores defined for the comparative cube space.

Example 23 (Comparative cube). The following comparative cube lists for each rural district of insurants in Upper Austria the ratio of drug costs in year 2012 to drug costs in year 2011, thereby comparison predicate SameDistrict is used to relate facts where PoI and PoC of the fact refer to the same district in the insurant dimension role.

(⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩
ins:ruralDistrict[time:top, ins:district, leadDoc:top, actDoc:top, drugATC:top]
SameDistrict(PoI.ins,PoC.ins)
⟨time:2011, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩
ins:ruralDistrict[time:top, ins:district, leadDoc:top, actDoc:top, drugATC:top])
..RatioOfDrugCosts

Measure and score instructions are arithmetic or aggregation expressions over measures of selected facts of measure cubes. The scoring instructions are either given in native form by using artihmetic and aggregation operations with MDO-query expressions (using '.' and '..'-operators as described above) as operands, or by SQL-built-ins. The measurement instruction of an *arithmetic measure* is

defined natively in MDO in an object-oriented flavor by an arithmetic expression over measures applied to a point self in the domain of the measure for which the measure value is calculated. The measurement instruction of an *aggregation measure* for point self is given in MDO by some kind of aggregation (such as AVG or SUM) over measure values of selected facts of some cube (which is a fact class for first-step aggregation measures). As the selection of facts is based on using concepts of the MDO as *qualifiers* to selecting facts (using ⟨self⟩(concept)-expressions for fact classes and ⟨self⟩(concept)[granularity]-expressions for other cubes; note: self may be omitted), we speak of ontology-based measures and scores.

Example 24 (Measure with one step aggregation). DrugCosts is a derived measure of type float over dimension space DrugPrescriptionSpace

```
CREATE MEASURE DrugCosts
DATATYPE float FOR DrugPrescriptionSpace AS
   SUM(⟨self⟩..drugPrescription.costs);
```

The measurement instruction indicates that the measure value is calculated for a point self by the sum over base measure costs of all facts in factclass drugPrescription that roll-up to self. If one restricts the dimension space DrugPrescriptionSpace to oral antidiabetic drugs (InOAD), one can define another measure that returns overall costs for oral antidiabetic drugs:

```
CREATE MEASURE OADDrugCosts
DATATYPE float FOR DrugPrescriptionSpace AS
   SUM(⟨self⟩InOAD..drugPrescription.costs);
```

Example 25 (Measure with two step aggregation). AvgDrugCostsPerIns is defined as a measure with two step aggregation that returns the average drug costs per insurant:

```
CREATE MEASURE AvgDrugCostsPerIns
DATATYPE float FOR DrugPrescriptionSpace AS
   AVG( ⟨self⟩[time:top, ins:insurant, leadDoc:top, actDoc:top, drugATC:top]
            ..DrugCosts );
```

The measure calculates, first, the total costs per insurant stated as cube ⟨self⟩ [time:top, ins:insurant, leadDoc:top, actDoc:top, drugATC:top]..DrugCosts (first aggregation step, cf. Ex. 24) and, second, the average of drug costs per insurant (second aggregation step).

Example 26 (Drill across measure). TotalCosts are computed by adding DrugCosts, which has been defined over dimension space DrugPrescriptionSpace, and AmbTreatmentCosts, which has been defined over dimension space AmbTreatmentSpace:

```
CREATE MEASURE TotalCosts
DATATYPE float FOR MedcareSpace AS
      ⟨self⟩.DrugCosts\0 + ⟨self⟩.AmbTreatmentCosts\0;
```

This is an example where two fact classes (drugPrescription and ambTreatment) are used, one providing measure DrugCosts and the other providing measure AmbTreatmentCosts. The dimension space MedcareSpace is defined as drill across space of DrugPrescriptionSpace and AmbTreatmentSpace (not shown). The decoration \0 indicates that the default value 0 is to be used if a measure application returns a null value.

Measurement and scoring instructions frequently have the same structure (or pattern). To avoid the need to define measurement and scoring instructions of similar kind, semCockpit provides a set of predefined measure and score templates, which can be extended. For example, the *first-step aggregation template* defines an aggregate measure based on the template parameters base measure m, slice-concept c, and aggregation function f, with underlying measurement instruction $f\langle\text{self}\rangle(c).m$ The *higher-step aggregation template* defines an aggregate measure based on: roll-up granularity g, derived measure m, and aggregation function f with underling measurement instruction $f\langle\text{self}\rangle[g].m$.

Scores are distinguished into arithmetic and analytic. Arithmetic scores relate two measures of two points (PoI and PoC) of a cube by an arithmetic function such as ratio or percentage difference.

Example 27 (Ratio score). Score RatioOfDrugCosts returns a ratio of drug costs:

```
CREATE SCORE RatioOfDrugCosts
DATATYPE float FOR (DrugPrescriptionSpace, DrugPrescriptionSpace) AS
    RATIO( ⟨PoI⟩.DrugCosts, ⟨PoC⟩.DrugCosts );
```

The score is defined for comparison dimension space (DrugPrescriptionSpace, DrugPrescriptionSpace). The keyword RATIO is used to indicate that drug costs of the group of interest represented as first parameter (PoI) are to be divided by the drug costs of the group of comparison denoted as second parameter (PoC).

Analytic scores use an analytic scoring function (such as average-percentile rank or mean-percentile rank) on two sets of points, GoI and GoC, each identified by a ⟨pnt⟩(concept)[granularity]-expression.

Example 28 (Median percentile rank score). Score MPROfDrugCostsPerPatient has median percentile rank as a scoring function. The score can be applied in the time dimension roles on month or higher levels, and in dimension roles ins, leadDoc, and actDoc from level district to top.

```
CREATE SCORE MPROfDrugCostsPerPatient
DATATYPE float FOR (DrugPrescriptionSpace, DrugPrescriptionSpace)
AT ([time:month..top, ins:district..top, leadDoc:district..top,
        actDoc:district..top, drugATC:drug..top],
    [time:month..top, ins:district..top, leadDoc:district..top,
        actDoc:district..top, drugATC:drug..top]) AS
  MEDIAN_PERCENTILE_RANK(
      ⟨PoI⟩[time:top, ins:insurant, leadDoc:top, actDoc:top, drugATC:top]
          ..DrugCosts,
      ⟨PoC⟩[time:top, ins:insurant, leadDoc:top, actDoc:top, drugATC:top]
          ..DrugCosts );
```

Drug costs are computed per insurant for group of interest as well as group of comparison. Based on both groupings the median percentile rank is calculated.

Generic measures and *generic scores* avoid the need of repeated definition of measure and score instructions with different selection criteria (concepts) in place. They provide for flexibility in measure and score use, and they enable reasoning (about how measures relate) by providing common structures. Generic measures and scores have generic parameters (denoted by %name) for concepts, which we call *qualifiers* as they are used to qualify selection-expressions for facts of some cube. The domain of a generic parameter may be restricted to a listed set of concepts or to the set of concepts subsumed by a concept.

Example 29 (Generic measures and scores). We generalize the definition of measure DrugCosts of Example 24 and add a generic parameter %q. It is restricted to multi-dimensional concepts which are subsumed by DrugPrescriptionSpace (denoted by ↑):

```
CREATE MEASURE DrugCosts( %q ↑ DrugPrescriptionSpace )
DATATYPE float FOR DrugPrescriptionSpace AS
   SUM(⟨self⟩(%q)..costs);
```

Analogously one can define generic scores like RatioOfDrugCosts:

```
CREATE SCORE RatioOfDrugCosts( %qoi ↑ DrugPrescriptionSpace,
                               %qoc ↑ DrugPrescriptionSpace )
DATATYPE float FOR (DrugPrescriptionSpace, DrugPrescriptionSpace) AS
      RATIO( ⟨PoI⟩.DrugCosts(%qoi), ⟨PoC⟩.DrugCosts(%qoc) );
```

Generic parameters can be used in place of concepts of concept expressions (e.g., oldPatient AND %q) in measurement or scoring instructions. A generic measure (score) is instantiated by binding the generic parameter to a concept, giving a non-generic measure (score) by replacing the generic qualifiers accordingly in measurement (scoring) instructions.

Example 30 (Use of generic measures and scores). Instantiation of generic measure DrugCosts with actual qualifier InOAD gives the previously defined measure OADDrugCosts (Ex. 24):

```
⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩
   .DrugCosts( InOAD )
```

The following instantiation of generic score RatioOfDrugCosts binds %qoi to InStarterOAD and %qoc to InOAD. It returns the ratio of starter oral antidiabetic drug costs to oral antidiabetic drug costs:

```
(⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩,
⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩)
   .RatioOfDrugCosts(InStarterOAD, InOAD)
```

Notice that in this special case of comparative data analysis point of interest and point of comparison are equal and the only difference is in the qualifications of the score.

In order to provide for the definition of measures and scores which cannot be directly expressed in the semCockpit language, *built-in measures and scores* provide for an ad-hoc extension facility and for the possibility to define new templates for defining measure or scores. A built-in measure or score is defined by providing an SQL view that defines the function from point to measure value (from pair of point to score value, respectively). This approach is not novel, but common in Oracle where new cubes and measures can be derived from multi-granular cubes of other measures. However, different to Oracle, all MDO-concepts are available as SQL views as well and may be used in measure- and score definitions. Thereby, rather than writing complex selection query predicates for selecting tuples of some cube, these can be easily selected by a simple (natural) join between the SQL view of the concept and the cube. This provides for a simple, natural definition of the measurement or scoring instructions. Platform-dependent optimization (in consideration of the capabilities and limitations of query optimizers) is a separate issue. Built-in generic measures and generic scores are provided as macros with qualifiers (view names) as parameters.

Measures and scores are organized in a measure & score drivers hierarchy. The change of value of a measure may change the value of another measure or score, the change of value of a score may change the value of another score. The influence hierarchy known from definition of measures and scores or from background knowledge (e.g., relationships of base measures) is captured explicitly and used later as background knowledge to guide analysis processes.

5 Ontology-Based Comparative OLAP

In online analytical processing, OLAP operations slice, dice, drill-down and roll-up are applied to a data cube in order to navigate from one to another cuboid of the data cube. We sketch a possible extension of semCockpit for modeling and representing such analysis steps.

In the context of ontology-based comparative data analysis and in the presence of generic scores, a particular comparative cuboid, which we call *comparative analysis situation*, is described in the form ⟨pntGoI⟩conceptGoI[granGoI]joinCond ⟨pntGoC⟩conceptGoC [granGoC]..score(qGoI,qGoC) with variables for nodes of points (pntGoI, pntGoC), levels of granularities (granGoI, granGoC), concepts (conceptGoI, conceptGoC, qGoI, qGoC), join condition (joinCond), and score. For simplicity, we consider here only scores with two qualifiers, one for the group of interest and one for the group of comparison.

The description of a non-comparative analysis situation is based on cube definition of the form ⟨point⟩concept [granularity]..measure (qualifier$_1$, ..., qualifier$_n$).

In the remainder, we speak for simplicity of analysis situations, if we refer to comparative and non-comparative analysis situations.

An OLAP-operation between two analysis situations, which we refer to as *navigation*, reflects the change of the bindings of the variables of the target analysis situation with respect to the source analysis situation. An atomic navigation changes the binding of a single variable. Atomic navigation can be classified in

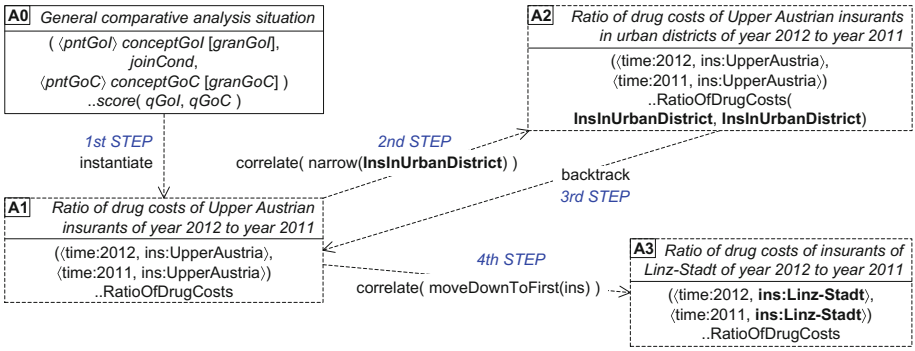

Fig. 10. Ontologoy-based comparative OLAP

an OLAP-setting according to the kind of OLAP-step performed. Such a *navigation step* indicates a movement along some "semantic relationship" between two cubes (such as drill-down one level in hierarchy x, move up to ancestor node in hierarchy x) and is expressed by a *navigation operator* and possibly a *navigation variable*. A variable may be for a node, a level, a concept, a join condition, or a score. The binding may be indicated absolute (i.e., by a new value) or relative to the binding of a source variable (e.g., drill-down one level from current level in hierarchy x).

Example 31 (Ontology-based comparative OLAP). Figure 10 illustrates[5] how a business analyst applies ontology-based comparative OLAP operators. First, A0 shows the general description of an analysis situation. She or he selects the points ⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩ and ⟨time:2011, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩ to compare drug costs of Upper Austria in year 2012 with 2011 by calculating the ratio (resulting in analysis situation A1). Next, the analyst considers that it is of interest to restrict the comparison to patients of urban districts. He or she applies the navigation operator correlate(narrow) with actual parameter InsInUrbanDistrict which narrows the group of interest as well as the group of comparison to urban districts. The navigation results in analysis situation A2 in which the generic score RatioOfDrugCosts is qualified in the GoI and in the GoC by hierarchical concept InsInUrbanDistrict. Afterwards the analyst backtracks to A1 and applies navigation operator correlate(moveDownToFirst) that results in moving down to the first district of Upper Austria in GoI and GoC (analysis situation A3). We assume a descending order by number of inhabitants, thus the user navigates to district Linz-Stadt.

A *composite analysis situation* is a tree of analysis situations and navigation steps. Composite analysis situations provide a global coherent picture of several

[5] In this and subsequent figures we omit for brevity the representation of "all"-nodes of points in dimension space DrugPrescription.

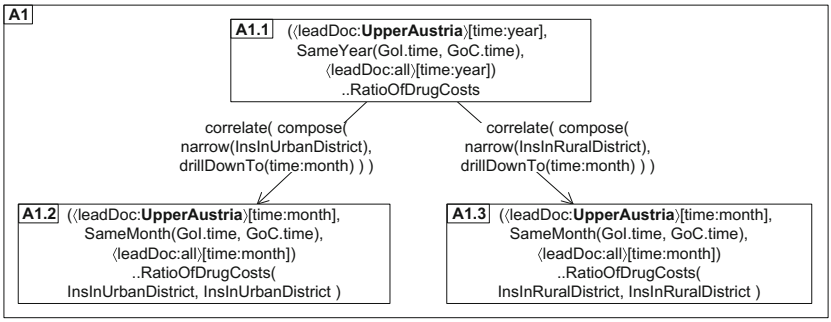

Fig. 11. Composite analysis situation

related measures and scores, aggregated and detailed. Changing the variables of the root analysis situation leads to coherent change of all dependent analysis situations. A generic composite analysis situation with all variables unconstraint and modifiable provides for a general, comprehensive multi-perspective browsing facility similar as it is provided by "surf and save" BI tools like Tableau[6], but with enhanced flexibility and guidance support (as will be discussed in the subsequent section). This mode of use is typical during phases of explorative search for meaningful scores and comparison groups, as well as for developing analysis processes (which are thereafter captured as more elaborated BI analysis graphs).

Example 32 (Composite analysis situation). By a composite analysis situation one can synchronize various semantically related analysis situations. In Fig. 11 root situation A1.1 selects drug costs ratios of lead doctors of province Upper Austria per year[7]. Correspondingly, A1.2 and A1.3 select the drug costs ratios of urban and rural districts, respectively, of Upper Austria per month. Two navigation operations denote these semantic relations. If the user changes lead doctor location from Upper to Lower Austria in A1.1, also the situations A1.2 and A1.3 are adapted automatically to province Lower Austria.

One can capture the history of a particular analysis performed, in the form of a graph, consisting of the analysis situations and navigation steps used to navigate between them.

Example 33 (History of an analysis). The essence of a history of an analysis can be depicted as a graph. Figure 12 presents the static view of the dynamic process of example 31. There are navigation arcs from analysis situation A1 to analysis situation A2 and A3. The graph only shows the semantic dependencies. It omits the dynamic behaviour like the order in which analysis situations were performed, or backtracking paths.

[6] http://www.tableausoftware.com

[7] In this and subsequent figures we omit for brevity the representation of "top"-levels of a granularity in dimension space DrugPrescriptionSpace.

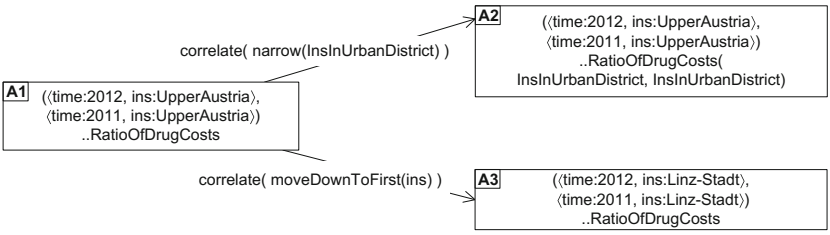

Fig. 12. History of an analysis (without backtracking steps)

6 BI Analysis Graphs

A typical BI analysis session in comparative data analysis is described by the following process: (1) Determine an initial analysis situation (by setting parameters of an OLAP query generating a cube or comparative cube, usually monogranular). (2) Visually inspect the result. (3) Modify parameters (usually based on semantic relationships) to move to a new analysis situation. (4) Inspect the result: (a) stop if satisfied, (b) continue with (3), or (c) backtrack to a previous analysis situation.

We present a vision of BI analysis graphs to capture this "analysis process knowledge" at the schema level for later analysis as reference. BI analysis graphs may be compared to process schemas designed in BPMN [37] which reflect how a particular kind of business process is handled, or to the navigation model of WebML [7] in web engineering, which describes how data and their relationships may be traversed.

BI analysis graphs are inspired by WebML: The navigation model of WebML is a graph of units and links. A unit represents objects or set of objects that are retrieved by a parameterized SQL query associated with the unit. Links describe how objects of source and target units relate. Changing the parameters of a source unit and, thus, the object(s) represented, is propagated to the target unit by information transported along the link (which binds parameters of source to parameters of target units), leading to related changes of the objects represented in the target unit. Such changes may be automatic or dependent on user input. Similarly, in BI analysis graphs, analysis situations are parameterized cube definitions (or MDO queries), and navigation steps between analysis situations bind parameters of the target based on parameters of the source and, optionally, dependent on user input. Different to BPMN and WebML, which come with a clear distinction between schema and instance, BI analysis graphs adhere to a Frame-inspired [9] approach in which generic and individual analysis situations co-exist in one graph. This reflects the very nature of BI analysis in which the BI analysis graph should on the one hand generalize from individual analysis, but on the other hand is never complete and as such is continually extended and refined based on known analysis process knowledge acquired in subsequent analysis.

Further related work concerns navigation modeling and query prediction [35], describing analytical sessions [33], modeling OLAP behavior [44], modeling preferences [13], personalization [2], query recommendations [3,11,19], and annotations [10]. Heer et al. [14] focus on recording and visualizing interaction histories and propose a taxonomy of interactive dynamics for visual analysis [15]. Thollot [43] propose a graph-based approach for context-aware BI recommendations. Unlike [14,15,43], BI analysis graphs are motivated for designing and re-using general (non-personalized) analysis processes, and for modeling navigation knowledge as semantic relationships (*'Navigation is Knowledge'*). Recording the history of an analysis process is a side product of the BI analysis graph approach. BI analysis graphs and the associated envisioned guidance rules (discussed in Sect. 7) draw ideas from active data warehouses [42] and from OLAP querying at a conceptual level [30]. A simple form of BI analysis graphs [26] has been introduced for multi-dimensional navigation modeling.

We now give an overview of BI analysis graphs. A BI analysis graph is a directed graph with generic or individual comparative analysis situations as vertices and generic navigation steps as directed edges (we will extend this definition later by specialization and instantiation edges).

A *generic comparative analysis situation* is a comparative analysis situation in which some or all variables are unbound and in which unbound variables are restricted by domain indications. The domain of a variable is given by a dimensional concept for node variables of points (and, optionally, by a multi-dimensional concept for a point or a comparative concept for the point pair), a set of concepts for concept variables (expressed by enumeration or as $\uparrow c$ for the set consisting of concept c and all subsumed concepts), a set of levels for level variables, a set of join conditions for join-condition variables, and a set of scores for score variables (where $\uparrow s$ denotes score s and any score that directly or indirectly drives s in the score drivers hierarchy). If a domain is not explicitly indicated for a variable, the domain is any value possible for the variable's kind. A variable that is bound to a value or has only one permissive value is said to be *closed*, otherwise it is said to be *open*. The notion of a *generic non-comparative analysis situation* is analogously defined.

Example 34 (Generic analysis situation). The top left corner of Fig. 13 shows a generic comparative analysis situation A0 where all variables for nodes of points (pntGoI, pntGoC), concepts of external slice conditions (conceptGoI, conceptGoC), levels of granularities (granGoI, granGoC), join condition (joinCond), score (score), and qualifiers (qGoI, qGoC) are open.

An *individual (comparative or non-comparative) analysis situation* is an analysis situation in which variables are bound. In a short hand notation, variables may not be used. In such a case, they are bound to default values (e.g., a missing granularity is bound to the granularity of the point). It is an *instance of* any generic (comparative or non-comparative, resp.) analysis situation for which all variable values are from the respective domains defined by the generic analysis situation.

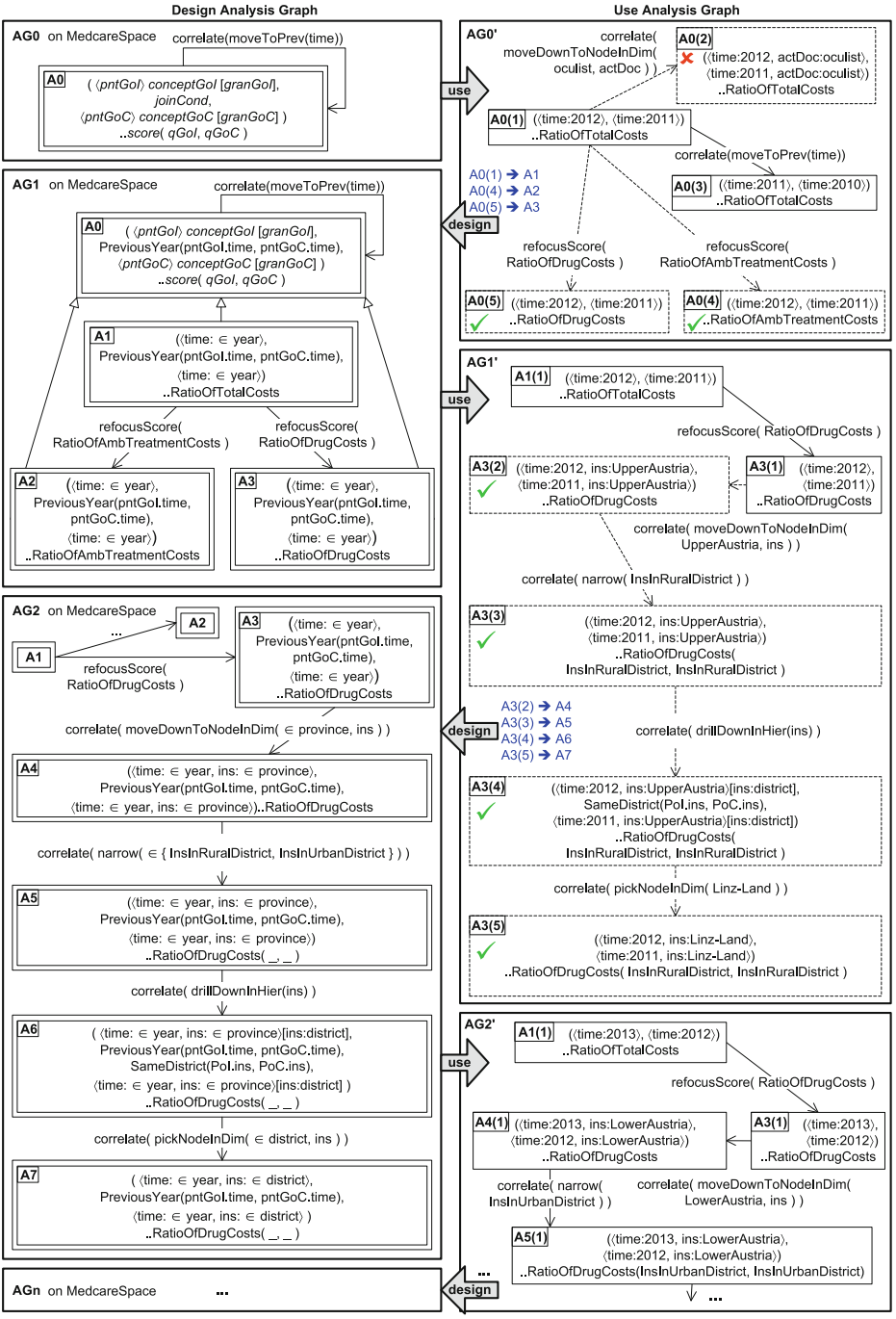

Fig. 13. BI Analysis Graph with use & design steps

In the remainder, we speak for simplicity of analysis situations, if we refer to comparative and non-comparative analysis situations, generic or individual.

Example 35 (Individual analysis situation). In Fig. 13 individual analysis situations are depicted on the right hand side. A business analyst uses the generic situation A0 and instantiates, e.g., the individual analysis situation A0(1).

A *generic navigation step* is defined by a navigation operator and, if it has a navigation variable, a domain for the navigation variable. An *individual navigation step* is given by applying the generic navigation step to an individual analysis situation and by binding any navigation variable of the navigation operator. The application yields an individual target analysis situations that is identical to the individual source analysis situation apart from the changes induced by the navigation operator.

Navigation operators express (a) movements in the nodes of a dimension hierarchy (like moveDownToNodeInDimension, moveUpInHierarchy, moveToPrev(ious sibling)), (b) changes of the granularity of a cube (like drillDownToLevel, drillDownInHierarchy, (c) strengthening (narrow), weakening (broaden), or resetting (qualifyAside) a qualifier of a score, (d) strengthening (filter), weakening (extend), or resetting (shift) the slice condition (i.e., conceptGoI or conceptGoC), or (e) the change of a score (refocus). Navigation operators may either change the group of interest (rerelate), the group of comparison (retarget), or both (correlate). Several navigation operators may also be composed into one (compose)). We omit a complete list for brevity.

Example 36 (Navigation step). Figure 13 shows a general navigation operation correlate(moveToPrev(time)) from A0 to itself. The instantiation of A0 to A0(1) and the application of the navigation step leads to A0(3).

We now revise the definition of BI analysis graphs to include modeled specialization relationships. A *BI analysis graph* is a directed graph whose vertices are analysis situations and whose directed edges are either navigation steps or modeled specialization relationships between analysis situations. Navigation steps may be specialized, too, in that a generic navigation step with a more restricted variable domain connects more specific analysis situations than the specialized navigation step. The specialization hierarchy is not inferred, but explicitly modeled similar to the conceptual modeling of business processes. It acts as constraint for the definition of analysis situations and allows to capture alternative navigation paths that apply to different specializations of a generic analysis situation. An analysis situation that *specializes* another analysis situation has for each variable the same or a more restrictive domain.

Navigation steps may be proper OLAP steps as introduced above or backtracking-steps. A backtracking step indicates that in a particular analysis the analyst moves back to a previously encountered analysis situation but chooses thereafter a different analysis situation to continue.

Example 37 (BI analysis graph). The left side of Fig. 13 shows analysis graphs AG0, AG1, and AG2. Each graph consists of vertices (analysis situations) and

directed edges (navigation steps). Backtracking-steps are not shown. Analysis situations A1, A2, and A3 are specializations of A0. Thus they are linked by inheritance arrows. Figure 13 is explained in more detail later.

A *BI analysis* is an alternate sequence of individual analysis situations and navigation steps that represent a sequential trace of the analysis steps performed by an analyst in a particular analysis. A particular BI analysis can be carried out by traversing an analysis graph and, if necessary, by extending the traversal.

A schema traversal $T = (A_1, S_1, \ldots, S_{k-1}, A_k)$ of a BI analysis graph G is an alternate sequence of analysis situations and navigation steps where for each $i = 1..k-1$ either (a) there exists an analysis situation A'_i such that S_i is an arc in G from A'_i to A_{i+1}, whereby $A'_i = A_i$, or A'_i is directly or indirectly connected to A_i by modelled specialisation relationships (or vice versa), or (b) S_i is a backtracking step to $A_{i+1} = A_j$ with $j < i$.

A BI analysis $t = (a_1, s_1, \ldots, s_{k-1}, a_k)$ is a *traversal* of a given BI analysis graph G if there exists a schema traversal $T = (A_1, S_1, \ldots, S_{k-1}, A_k)$ of G such that a_1 is an instance of A_1 and for $i = 1..k-1$, a navigation step S_i of G such that a_i is an instance of A_i, s_i is an individual navigation step of generic navigation step S_i, and a_{i+1} is an instance of A_{i+1}.

A traversal of a given BI analysis graph is specified by binding the open variables of some analysis situation of the BI analysis graph and by subsequent bindings of all open navigation variables of navigation steps followed. Notice that the definition of a traversal of a BI analysis graph permits to initially jump to any node of the graph and, once some analysis situation is reached, it does not require to continue with the most specific navigation step defined in the BI analysis graph. The graph acts as guidance and is not prescription, it can be incomplete. A traversal of a navigation step that has been already specialized may lead later to an inclusion of a different specialization in the BI analysis graph.

Example 38 (BI analysis). The right side of Fig. 13 demonstrates the use of analysis graphs AG0, AG1, and AG2, leading to BI analyses AG0', AG1', and AG2'. All open variables of analysis situations and navigation steps are bound. The sequence (A1(1), refocusScore (RatioOfDrugCosts), A3(1), ..., A3(5)) of analysis AG1' is a traversal of AG1.

Once an initial analysis graph has been defined, proper repeated analysis proceeds as follows: (1) Jump to a predefined initial analysis situation (2) Set open parameters for this analysis situation. (3) Evaluate the analysis situation and inspect the result. (4) Choose an outgoing arc to move to another analysis situation. (5) Set an open parameter for this arc, if any. (6) Evaluate and inspect, the result; stop if satisfied or continue with 4.

If the business analyst is not satisfied with the options provided by the BI analysis graph, he or she may add new edges and vertices that move to new terrain or specialize given edges or vertices (in that more parameters are bound and thus closed, or choices of bindings of open parameters are restricted). semCockpit provides reasoning support to determine applicable values for open parameters

and to check consistency of BI analysis graphs, especially with respect to node and link specialization. The whole analysis process can be described in alternating use- and design-phases, i.e., an analyst applies an existing analysis graph (use-phase) and, if necessary, extends or modifies it (design-phase), afterwards she uses the new graph, etc.

Example 39 (Analysis process). Figure 13 demonstrates the explorative, iterative, and incremental characteristics of an analysis process. A business analyst alternates between design and use of analysis graphs. When she or he uses the graph the analyst also performs individual explorations:

(1) Initial analysis graph: AG0 comprises a generic analysis situation A0. The navigation step from A0 to itself with navigation operation correlate(moveToPrev(time)) represents the rule of thumb that if some analysis situation is relevant then the similar analysis situation for the previous time period is also relevant.
(2) Use phase: A business analyst instantiates analysis graph AG0 by binding variables pntGoI.time and pntGoC.time to 2012 and 2011, respectively, resulting in analysis graph AG0' with individual analysis situation A0(1). Following the navigation step proposed in AG0, the business analyst moves to analysis situation A0(3). Further, the business analyst makes *exploratory* individual navigation steps, which are not proposed as such in AG0, leading to analysis situations A0(2), A0(4), and A0(5). Some of these comparisons are deemed relevant, A0(4) and A0(5), others are not, A0(2). In Fig. 13 relevant analysis situations are decorated by a hooklet and others are decorated by a cross. Analysis situations instantiated from generic ones, such as A0(1) and A0(3), are depicted as boxes with solid border. Analysis situations reached through exploratory navigation steps, such as A0(2), A0(4), and A0(5), are depicted as boxes with dotted border.
(3) Design phase: The business analysts wants to re-use analysis situations A0(1), A0(4), and A0(5) in later similar analysis sessions and generalizes them, in analysis graph AG1, to A1, A2, and A3, respectively. Variable pntGoI.time with domain year takes the place of constant 2012 and variable pntGoC.time takes the place of constant 2011. The relation between 2012 and 2011 is represented by constraining pntGoI and pntGoC to comparative concept PreviousYear. Analysis situations A1, A2, and A3 are specializations of A0, which is depicted by specialization links.
(4) Continuation of alternating use and design phases: The business analyst incrementally designs a BI analysis graph, from analysis graph AG1 to analysis AG1' to analysis graph AG2, and so forth. Resulting BI analysis graphs can be re-used in various related analyses, for example, to reiterate the analysis in the next year in order to reassess the conclusions made by the analyst or in order to monitor the effects of actions taken in reaction to the analysis.

7 Guidance, Judgement, and Analysis Rules

Guidance, judgement, and analysis rules provide actionable knowledge about comparative analysis that otherwise is tacit knowledge of a business analyst or captured and processed in some other form, usually outside BI analysis tools.

A simple form of actionable knowledge is supported in many BI systems by the possibility of setting alerters that fire if some measure exceeds a certain threshold. Guidance rules are defined over analysis situations and provide guidance on how to proceed best in analysis, by suggesting a generic or individual navigation step to follow or advising that a particular navigation is not deemed relevant. Judgement rules are defined over facts of a comparative cube and represent static knowledge about possible explanations of a striking low or high score. Analysis rules are defined over facts of a comparative cube as well and decide based on the score of a fact, whether a specific action, e.g., starting a specific analysis, should be taken (action rules), or whether a fact should be reported (reporting rules). Analysis rules are evaluated for specific set of point pairs explicitly identified (e.g. after an ETL cycle), and we will see later that they require different evaluation strategies in the context of inheritance and overriding.

A *guidance rule* is given by (a) a name, (b) an analysis situation (generic or individual), (c) a rule condition which is either (c1) a condition over all facts of the analysis situation (set-oriented rule) or (c2) a condition over a fact of the analysis situation (fact-oriented rule), and (d) a recommended or disadvised generic or individual navigation operation, whose variable domain may be restricted or its variable set (d1) absolute or (d2) relative to variables of the individual analysis situation (for set-oriented rules) or also of coordinates of the fact (for fact-oriented rules) for which the rule fires. In case of a composite analysis situation the guidance rule is defined for the root analysis situation and the rule conditions may also consider the component analysis situations and their facts. Rule conditions are expressed as SQL queries (set- or tuple-oriented) over the (comparative) cubes of the analysis situations. A set-oriented rule (FOR analysis situation ONCE) fires once for an individual analysis situation, if the set-oriented query is not null. A tuple-oriented rule (FOR analysis situation) fires for each fact that is retrieved by the SQL query.

Guidance rules of the same name are organized in an inheritance hierarchy based on specialization relationships between generic analysis situations for which they are defined.

A guidance rule *applies* to an analysis situation if it is an instance of the generic analysis situation for which the rule is defined and there is no more specific generic analysis situation with this property. If applicable, a set-oriented rule *fires* for an individual analysis situation if the rule-condition is satisfied; a fact-oriented rule fires for every fact of the individual analysis situation for which the rule-condition is satisfied, and, depending on the rule, the navigation is recommended (RECOMMEND) or disadvised (DISADVISE). Guidance rules are evaluated after every individual navigation step of a BI analysis.

Several guidance rules may apply to a given individual analysis situation. Since guidance rules suggest potentially promising navigation steps to follow in subsequent analysis, a unique choice is not required but rather it may be worthwhile to explore all options suggested to gain further insight about the data at hand. Inheritance as described above can be used to specialize guidance rules by providing more specific suggestions for more specific analysis situations.

More specific guidance rules override more general ones and, thus, exclude the more general suggestions from the suggestions list. Initially, very general guidance rules may be defined that are extended and specialized over time.

Example 40 (Guidance Rules). Figure 14 shows guidance, judgement, and analysis rules (action and reporting rules). The guidance rules refer to the analysis situation in Fig. 13, but guidance rules may be also defined for generic analysis situations independent of analysis graphs (not shown). GR0 for analysis situation A0 recommends navigation operation correlate(moveToPrev(time)), if the score value is less than 0.8 or greater than 1.2. For analysis situation A3, which is a specialization of A0, guidance rule GR0 for A0 is overridden.

A *judgement rule* is specified by (a) a name, (b) a comparative cube with a single score (defining the facts over which the judgement rule is defined), and (c) a score-value comparison over such a fact, and (d) an informative judgement. Judgement rules are evaluated any time a fact over which the judgement rule is defined is retrieved and fires if the condition is met. Judgement rules of the same name are organized in an inheritance hierarchy along the subset relationships of the set of point-pairs of comparative cubes over which they are defined. A generic judgement rule may be defined for a comparative cube with a generic score whereby the score qualifiers may be constrained in the join condition. A generic judgement rule inherits from another judgement rule if the set of point pairs of the former is a subset of the set of point pairs of the latter, both are for the same score with the same qualifiers, and the domains of the qualifiers are in subset relationships. A generic judgement rule is defined for each fact with point pairs in the comparative cube and with instantiations of the generic score whose actual qualifiers are in the qualifier domains.

Example 41 (Judgement Rules). Judgement rule AR1 for cube C1 in Fig. 14 indicates, once a fact of cube C1 is accessed, that one has to take into account an increase of drug costs of about 5 % per year. The rule is overridden for facts of cube C2, which specializes cube C1 (for year 2012), and justifies an exceptional increase for oral antidiabetic drugs in year 2012.

An *analysis rule* is specified like a judgement rule, but different to judgement rules its action is a recommended action or report and analysis rules need to be explicitly triggered (for example after an ETL-cycle has been completed for the set of new facts just loaded into the DWH) and analysis rules have two conditions (explained later). In its basic form, application and triggering, as well as inheritance is defined as for judgement rules. But, rather than guidance and judgement, which concern an individual analysis situation or fact at hand during a BI analysis, actions and reports are compiled for a bulk of facts where some additional form of abstraction and accompanying rule evaluation strategy is required to reduce repetition and information overload. Moreover, analysis rules with the same name will frequently be for the same action or report, and vice versa.

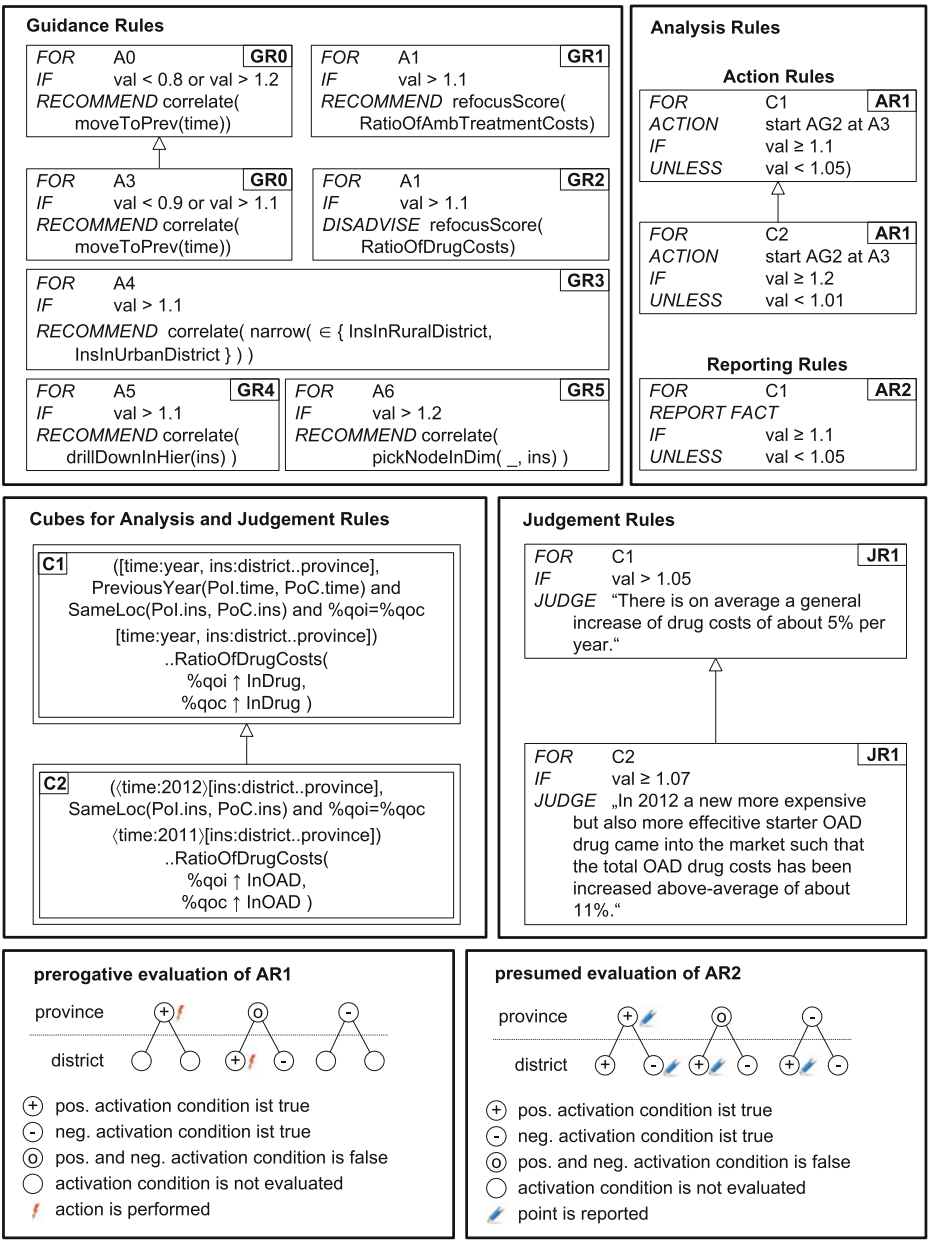

Fig. 14. Guidance, Judgement, and Analysis Rules

It is common in organizational contexts and in law to apply a decision-scope approach [36] to decision making. Higher organization levels set a decision scope within which lower organization levels may operate. In case of conflict the

regulations and rules of a higher level (e.g., European Union) take precedence over those of a lower level (e.g., a member state). Such rules define under which conditions certain actions may, must not, or need to be taken. This approach can also be applied to action and reporting rules with two conditions, one (IF condition or positive activation condition) stating when the rule should fire and the other one (UNLESS condition or negative activation condition) when the rule should not fire. If both conditions are not complementary, a decision scope for more specific rules is left.

Applying the decision scope approach to analysis rules in BI analysis requires to consider two alternative hierarchies: (1) hierarchies of sets of points at the same granularity and (2) the roll-up hierarchy of points in multi-dimensional space.

Specialization along subset relationships between sets of point at the same granularity is governed by the same rules for inheritance and overriding as introduced before. Specialization along a roll-up hierarchy of points in multi-dimensional space actually concerns entities of different kinds, yet connected by some form of part-of relationship. This gives rise to two alternative rule evaluation strategies, both meaningful in practice, but with a different area of application, actioning and reporting in mind.

We first consider the roll-up hierarchy of points (or pairs of points) and we will then discuss the interplay between specialization along subset relationships of points and the roll-up hierarchy of points. We discuss two evaluation strategies. The prerogative strategy which is more appropriate for action rules and the presumed strategy which is more appropriate for reporting rules.

In the *prerogative* strategy, an action triggered for a higher-level point (or pairs of points) implicitly implies the same action for each lower level point (or pairs of points). E.g., if a company decides to abandon a product line (such as mobile phones) this decision is implied for every product (i.e., every phone model in our example) of this product line. In the prerogative evaluation strategy, analysis rules for the same action are evaluated top-down along a user-specified roll-up path of granularities. We assume at first that for points of one granularity at most one analysis rule is defined with a positive and negative activation condition. If one of the two condition applies for a roll-up fact, analysis is completed, whereby the indicated action is triggered, if the positive activation condition is satisfied. Only if both conditions are not satisfied, i.e., for the situation that a fact falls into the open space of the condition scope, the analysis rules for drill-down granularities and drill-down facts at these granularities are considered in further rule evaluation.

Example 42 (Action rules – prerogative evaluation strategy). In Fig. 14 action rule AR1 is defined for facts in cube C1. Remember that action rules need to be explicitly evaluated for a specific set of facts (e.g., those just loaded into the data warehouse), identified by multi-granular cube and for a roll-up path of granularities. This roll-up path needs not to be indicated, if the analysis rule is only over multi-granular cubes with granularities along a single roll-up path, which is the case for AR1. If rule AR1 fires, it starts a traversal of analysis graph

AG2 at situation A3 of Fig. 13. For AR1 we have a prerogative evaluation strategy. If the positive activation condition ($val \geq 1.1$) is true for a province, AG2 is started for that province. If neither the positive ($val \geq 1.1$) nor the negative ($val < 1.05$) activation condition is true, the rule evaluates the conditions for districts to decide whether an analysis graph traversal is triggered.

In the *presumed* strategy, a report triggered for a higher-level point is presumed to cover also lower-level points and is thus not reported again for lower-level points (to avoid unnecessary information overload, the basic motivation behind performing roll-up analysis in data warehousing), unless a lower-level point fulfills the negative activation condition of a more specific rule. E.g., if a reporting rule triggers the report that average treatment costs for patients with diabetes mellitus of type 2 patients in Austria are twice as high than in Germany last year, it is presumed that this will hold in general for each province in Austria. Minor deviations are generally not of interest in comparative data analysis, but major ones are. The knowledge what kind of exceptions should be reported can be expressed by a negative activation condition of a more specific rule. In our example such a rule may state that for comparing treatment costs for patients in a province of Austria with patients in Germany, a percentage difference of less than 20 % should not be reported. Assume a positive activation condition of a more general rule has led to report a striking difference of average treatment costs per patient in Austria versus Germany. Based on deviating observation for comparing Tyrol (a province of Austria) with Germany the more specific rule will report (if the difference is less than 20 %) that contrary to Austria as a whole, average treatment costs per patient have been similar when comparing Tyrol with Germany.

Example 43 (Reporting rules – presumed evaluation strategy). AR2 of Fig. 14 represents a reporting rule, which is evaluated in a presumed manner. If the positive activation condition ($val \geq 1.1$) is true for a province, the province is reported, but districts of the province are only reported, if the evaluation of the negative activation condition ($val < 1.05$) is true for a district of that province. In this case the district is reported as an exception.

Prerogative and presumed evaluation of analysis rules along roll-up hierarchies in top-down manner can be combined with evaluation along hierarchies of sets points at the same granularity according to the specialization principle described above for guidance rules. Again we assume for simplicity, that analysis rules are triggered for points along a single-drill-down path of granularities. At each granularity all analysis rules defined for points at that granularity are considered, and for each fact the most specific one is chosen (It is assumed here, as that the specialization hierarchy is consistent such that a single most specific rule exists). We also expect that the rules have been specialized in a consistent way according to the decision scope approach such that the positive activation condition of a more general rule implies the activation condition of a more specific rule, and the same holds for negative activation rules. In the prerogative

strategy, once a decision has been determined for a fact at a given granularity, analysis stops; if no decision could be made due to an open decision scope, analysis continues with drill-down facts at a lower-level granularity for which an analysis rule is defined. In the presumed evaluation strategy, a reporting rule that once fired for a fact, is not triggered again for drill-down facts to avoid information overload, unless reporting an exception is justified by a negative activation condition.

Example 44 (Specialization of an action rule). In Fig. 14 action rule AR1 for cube C1 is overridden by the same named action rule AR1 for cube C2. If the rule is applied to a multi-dimensional point with year 2012, the specialized rule AR1 for cube C2 is evaluated.

8 Conclusion

Existing BI tools are well-suited for reporting and for performing complex analysis tasks but lack an explicit formalization of knowledge about business terms, comparison, and analysis processes. Commonly, business analysts define business terms in an ad-hoc manner rather than explicitly capturing business terms and their meaning in a central, shared repository. An unambiguous formalization of business terms can be used for the definition of measures, which facilitates the analysis task of a business analyst. Furthermore, the definition of meaningful comparisons should not be left to human intuition but captured as a first-class citizen. Similarly, the analysis process itself could be formalized. The experience of business analysts should be made explicit in order to benefit all business analysts within a company.

Rather than solving each issue separately, we presented an integrated and coherent approach for ontology-driven comparative data analysis.

The centerpiece of the Semantic Cockpit approach is the multi-dimensional ontology (MDO) which enriches a data warehouse with a set of concepts. These MDO concepts unambiguously define business terms and their meaning in the context of data analysis. Existing domain ontologies can be integrated as semantic dimensions. Analysts use MDO concepts for the definition of measures and scores. Measures quantify real-world facts of interest. Scores contrast measure values of a group of interest with a comparison group. Generic measures and scores reduce the number of measures that have to be defined. Judgement rules provide possible explanations for striking results that are encountered during an analysis process. Analysis rules trigger particular analysis processes or report facts.

The semCockpit project extends the enterprise data warehouse by a static semantic layer which assists analysts in formulating analytical queries in high-level business terms. During the project we identified the need to also capture knowledge about analysis processes. Future research concerns modeling and sharing of such knowledge in a semantic BI process layer based on the ideas of analysis graphs and guidance rules sketched herein.

Acknowledgments. This work is funded by the Austrian Ministry of Transport, Innovation, and Technology in program FIT-IT Semantic Systems and Services under grant FFG-829594 (*Semantic Cockpit:* an ontology-driven, interactive business intelligence tool for comparative data analysis).

References

1. Anderlik, S., Neumayr, B., Schrefl, M.: Using domain ontologies as semantic dimensions in data warehouses. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 88–101. Springer, Heidelberg (2012)
2. Bellatreche, L., Giacometti, A., Marcel, P., Mouloudi, H., Laurent, D.: A personalization framework for OLAP queries. In: Song, I.-Y., Trujillo, J. (eds.) DOLAP, pp. 9–18. ACM, New York (2005)
3. Bentayeb, F., Favre, C.: RoK: roll-up with the K-means clustering method for recommending olap queries. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 501–515. Springer, Heidelberg (2009)
4. Berger, S., Schrefl, M.: Feddw: A tool for querying federations of data warehouses - architecture, use case and implementation. In: Cordeiro, J., Filipe, J. (eds.) ICEIS (1), pp. 113–122 (2009)
5. Buchheit, M., Nutt, W., Donini, F.M., Schaerf, A.: Refining the structure of terminological systems: Terminology = schema + views. In: Hayes-Roth, B., Korf, R.E. (eds.) AAAI, pp. 199–204. AAAI Press/The MIT Press (1994)
6. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The mastro system for ontology-based data access. Semant. Web **2**(1), 43–53 (2011)
7. Ceri, S., Brambilla, M., Fraternali, P.: The history of webML lessons learned from 10 years of model-driven development of web applications. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 273–292. Springer, Heidelberg (2009)
8. Das, S., Chong, E.I., Eadon, G., Srinivasan, J.: Supporting ontology-based semantic matching in rdbms. In: Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B. (eds.) VLDB, pp. 1054–1065. Morgan Kaufmann (2004)
9. Fikes, R., Kehler, T.: The role of frame-based representation in reasoning. Commun. ACM **28**(9), 904–920 (1985)
10. Geerts, F., Kementsietsidis, A., Milano, D., et al.: *i*MONDRIAN: a visual tool to annotate and query scientific databases. In: Böhm, C. (ed.) EDBT 2006. LNCS, vol. 3896, pp. 1168–1171. Springer, Heidelberg (2006)
11. Giacometti, A., Marcel, P., Negre, E., Soulet, A.: Query recommendations for OLAP discovery driven analysis. In: Song, I.-Y., Zimányi, E. (eds.) DOLAP, pp. 81–88. ACM (2009)
12. Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: a conceptual model for data warehouses. Int. J. Coop. Inf. Syst. **7**(2–3), 215–247 (1998)
13. Golfarelli, M., Rizzi, S., Biondi, P.: myOLAP: an approach to express and evaluate OLAP preferences. IEEE Trans. Knowl. Data Eng. **23**(7), 1050–1064 (2011)
14. Heer, J., Mackinlay, J.D., Stolte, C., Agrawala, M.: Graphical histories for visualization: supporting analysis, communication, and evaluation. IEEE Trans. Vis. Comput. Graph. **14**(6), 1189–1196 (2008)
15. Heer, J., Shneiderman, B.: Interactive dynamics for visual analysis. Commun. ACM **55**(4), 45–54 (2012)

16. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation, 27 October 2009. http://www.w3.org/TR/owl2-primer/
17. Hürsch, W.L.: Should superclasses be abstract? In: Pareschi, R. (ed.) ECOOP 1994. LNCS, vol. 821, pp. 12–31. Springer, Heidelberg (1994)
18. Hurtado, C.A., Mendelzon, A.O.: Reasoning about summarizability in heterogeneous multidimensional schemas. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, p. 375. Springer, Heidelberg (2001)
19. Jerbi, H., Ravat, F., Teste, O., Zurfluh, G.: Preference-based recommendations for OLAP analysis. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 467–478. Springer, Heidelberg (2009)
20. Khouri, S., Bellatreche, L.: A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In: Song, I.-Y., Ordonez, C. (eds.) DOLAP, pp. 19–24. ACM (2010)
21. Lehner, W., Albrecht, J., Wedekin, H.: Normal forms for multidimensional databases. In: Rafanelli, M., Jarke, M. (eds.) SSDBM, pp. 63–72. IEEE Computer Society (1998)
22. Lim, L., Wang, H., Wang, M.: Unifying data and domain knowledge using virtual views. In: Koch, C., Gehrke, J., Garofalakis, M.N., Srivastava, D., Aberer, K., Deshpande, A., Florescu, D., Chan, C.Y., Ganti, V., Kanne, C.-C., Klas, W., Neuhold, E.J. (eds.) VLDB, pp. 255–266. ACM (2007)
23. Malinowski, E., Zimányi, E.: Hierarchies in a multidimensional model: from conceptual modeling to logical representation. Data Knowl. Eng. **59**(2), 348–377 (2006)
24. Nebot, V., Llavori, R.B.: Building data warehouses with semantic web data. Decis. Support Syst. **52**(4), 853–868 (2012)
25. Nebot, V., Berlanga, R., Pérez, J.M., Aramburu, M., Pedersen, T.B.: Multidimensional integrated ontologies: a framework for designing semantic data warehouses. In: Spaccapietra, S., Zimányi, E., Song, I.-Y. (eds.) Journal on Data Semantics XIII. LNCS, vol. 5530, pp. 1–36. Springer, Heidelberg (2009)
26. Neuböck, T., Neumayr, B., Rossgatterer, T., Anderlik, S., Schrefl, M.: Multidimensional navigation modeling using BI analysis graphs. In: Castano, S., Vassiliadis, P., Lakshmanan, L.V.S., Lee, M.L. (eds.) ER Workshops 2012. LNCS, vol. 7518, pp. 162–171. Springer, Heidelberg (2012)
27. Neumayr, B., Schrefl, M., Thalheim, B.: Hetero-homogeneous hierarchies in data warehouses. In: Link, S., Ghose, A. (eds.) APCCM. CRPIT, vol. 110, pp. 61–70. Australian Computer Society (2010)
28. Neumayr, B., Schütz, Ch., Schrefl, M.: Semantic enrichment of OLAP cubes: multidimensional ontologies and their representation in SQL and OWL. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) OTM 2013. LNCS, vol. 8185, pp. 624–641. Springer, Heidelberg (2013)
29. Niinimäki, M., Niemi, T.: An etl process for olap using rdf/owl ontologies. In: J. Data Semantics [39], pp. 97–119
30. Pardillo, J., Mazón, J.-N., Trujillo, J.: Extending OCL for OLAP querying on conceptual multidimensional models of data warehouses. Inf. Sci. **180**(5), 584–601 (2010)
31. Romero, O., Abelló, A.: Automating multidimensional design from ontologies. In: Song, I.-Y., Pedersen, T.B. (eds.) DOLAP, pp 1–8. ACM (2007)
32. Romero, O., Abelló, A.: Open access semantic aware business intelligence. In: Zimányi, E. (ed.) eBISS 2013. LNCS, vol. 7911, pp. xx–yy. Springer, Heidelberg (2014)

33. Romero, O., Marcel, P., Abelló, A., Peralta, V., Bellatreche, L.: Describing analytical sessions using a multidimensional algebra. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 224–239. Springer, Heidelberg (2011)
34. Saggion, H., Funk, A., Maynard, D., Bontcheva, K.: Ontology-based information extraction for business intelligence. In: Aberer, K. (ed.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 843–856. Springer, Heidelberg (2007)
35. Sapia, C.: On modeling and predicting query behavior in OLAP systems. In: Gatziu, S., Jeusfeld, M.A., Staudt, M., Vassiliou, Y. (eds.) DMDW. CEUR Workshop Proceedings, vol. 19, pp. 1–10. CEUR-WS.org (1999)
36. Schrefl, M., Neumayr, B., Stumptner, M.: The decision-scope approach to specialization of business rules: Application in business process modeling and data warehousing. In: Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling (APCCM 2013) (2013)
37. Silver, B.: BPMN Method and Style, 2nd edn., with BPMN Implementer's Guide: A Structured Approach for Business Process Modeling and Implementation Using BPMN 2.0. Cody-Cassidy Press (2011)
38. Skoutas, D., Simitsis, A., Sellis, T.K.: Ontology-driven conceptual design of etl processes using graph transformations. In: J. Data Semantics [39], pp. 120–146
39. Spaccapietra, S., Zimányi, E., Song, I.-Y. (eds.): Journal on Data Semantics XIII. LNCS, vol. 5530. Springer, Heidelberg (2009)
40. Spahn, M., Kleb, J., Grimm, S., Scheidl, S.: Supporting business intelligence by providing ontology-based end-user information self-service. In: Duke, A., Hepp, M., Bontcheva, K., Vilain, M.B. (eds) OBI. ACM International Conference Proceeding Series, vol. 308, p. 10. ACM (2008)
41. Staudt, M., Jarke, M., Jeusfeld, M.A., Nissen, H.W.: Query classes. In: DOOD, pp. 283–295 (1993)
42. Thalhammer, T., Schrefl, M., Mohania, M.K.: Active data warehouses: complementing OLAP with analysis rules. Data Knowl. Eng. **39**(3), 241–269 (2001)
43. Thollot, R.: Dynamic Situation Monitoring and Context-Aware BI Recommendations. PhD thesis, Ecole Centrale Paris (2012)
44. Trujillo, J., Gómez, J., Palomar, M.S.: Modeling the behavior of OLAP applications using an UML compliant approach. In: Yakhno, T. (ed.) ADVIS 2000. LNCS, vol. 1909, pp. 14–23. Springer, Heidelberg (2000)