Guided Query Composition with Semantic OLAP Patterns*

Ilko Kovacic
Johannes Kepler University Linz
Linz, Austria
ilko.kovacic@jku.at

Christoph G. Schuetz
Johannes Kepler University Linz
Linz, Austria
christoph.schuetz@jku.at

Simon Schausberger
Johannes Kepler University Linz
Linz, Austria
simon.schausberger@jku.at

Roman Sumereder
Johannes Kepler University Linz
Linz, Austria
roman.sumereder@jku.at

Michael Schrefl
Johannes Kepler University Linz
Linz, Austria
michael.schrefl@jku.at

ABSTRACT

Enabling domain experts to independently compose ad hoc OLAP queries is the primary goal of semantic OLAP (semOLAP) patterns. In this respect, a semOLAP pattern represents a recurring domain-independent OLAP query by describing the application scope and defining the structure of the query using formal pattern elements (FPEs). Such a semOLAP pattern is executable: In order to execute a semOLAP pattern, the user instantiates the pattern by providing FPE bindings. In this paper, we propose an approach for guided query composition which considers the inherent query structure in order to determine a navigation flow and recommend possible bindings for the corresponding FPEs. Guidance supports both existing as well as future, currently unidentified semOLAP patterns. The presented approach has been implemented in the course of a collaborative research project between industry and academia on precision dairy farming.

1 INTRODUCTION

Data warehousing and online analytical processing (OLAP) facilitate data-driven decision making, allowing domain experts to make rational decisions. A data warehouses organizes data in a multidimensional space (*data cube*). Each point in such a multidimensional space represents an occurrence of a business event (*fact*) which is quantified by measures. Hierarchically organized dimensions support the aggregation of facts along a hierarchy of granularity levels, e.g., day to month, city to county.

Standardized reports provide access to data warehouses in order to satisfy the domain expert's information needs. These reports are usually not static but rather support the specification of selection criteria restricting only one dimension (*slice*) or multiple dimensions (*dice*). Each report executes a predefined underlying query – an OLAP query – to retrieve the required information. Reports, however, can only satisfy about 60-80% of the information needs [6, p. 19]. Satisfying the remaining information needs requires the composition of ad hoc OLAP queries.

In order to compose ad hoc OLAP queries, domain experts must have knowledge about the underlying schema and the employed query language. Domain experts, however, typically lack the required knowledge and, therefore, must rely on assistance for ad hoc OLAP query composition.

In the course of several industrial research projects, such as semCockpit [9] and agriProKnow [13], we have noticed that, independent of a specific domain, ad hoc OLAP queries follow certain recurring patterns. In previous work we have hence identified and semantically described those patterns, leading to semantic OLAP (semOLAP) patterns (for details see [10]). A semOLAP pattern comprises a structural definition using formal pattern elements (FPEs) as well as a textual description including a concise name, the analysis situation that the pattern can be applied in, the instructions to follow, and an example.

The semOLAP pattern approach is followed in the agriProKnow project to support precision diary farming. The aim of precision dairy farming is to exploit data generated by agricultural cyber-physical systems to improve the overall health of the herd through early diagnosis and prevention of diseases [2]. In the course of the agriProKnow project animals are tracked by milk robots measuring milk yield and milk components, veterinarians capturing the animals' health state, smart ear tags tracking animal movement, and micro-climate sensors capturing environmental conditions. These data are transformed and loaded into a data warehouse which allows to compare animals across different farm sites. The data warehouse is accessed by domain experts such as veterinarians and farmers, allowing data-driven decision making. For example, a domain expert who wants to compare the milk yield of all young cows with the milk yield of all cows of the farm site Kremesberg per date starts with the selection of a suitable semOLAP pattern, i.e., the homogeneous set-base comparison pattern which allows to compare a subset with its base set. The selected pattern is then instantiated by considering the domain expert's information need. Finally, an OLAP query is generated based on a pattern instance in order to retrieve the required information (see Fig. 1).

The domain expert provides values for a semOLAP pattern's FPEs during the instantiation. The values provided for the FPEs must satisfy constraints regarding the pattern structure, the schema, and existing FPE bindings. To guide the domain expert in this difficult task, an interactive interface is required which allows to refine the semOLAP pattern instance by navigating from one FPE to another. A domain expert should receive recommendations of possible FPE value bindings to ease the process of instantiation and to avoid the generation of possibly invalid queries by enforcing existing constraints. The guidance approach should facilitate instantiation of all semOLAP patterns, even those which are not yet defined. Therefore, the approach should be specific enough to incorporate pattern-specific characteristics yet general enough to support all existing and future semOLAP patterns.

*This research was conducted as part of the agriProKnow project (http://www.agriProKnow.com/), funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "Production of the Future" between 11/2015 and 01/2018, Grant No. 848610.

In this paper we propose a guidance approach for semOLAP pattern instantiation. After selection of a semOLAP pattern, the user is guided through the steps of the instantiation process following a navigation flow. This navigation flow connects all activities required to instantiate the FPEs. For each FPE, possible bindings are recommended to the user. To this end, the guidance approach relies on a semOLAP knowledge graph, which contains knowledge about the pattern structure, the underlying schema, and the bindings of already instantiated FPEs. The semOLAP knowledge graph enables the recommendation of bindings which are suitable for specific FPEs. The user can select values as bindings during the instantiation of the pattern without deeper knowledge of the underlying schema, dependencies between query elements, and the query language. After all FPE bindings are specified by the user, the instantiation process is finished and a corresponding query is generated in order to retrieve the required information. The implementation of the data warehouse employs a relational database where different fact classes are stored as fact tables. The semOLAP knowledge graph is represented in Resource Description Framework (RDF) format. The interaction flow modelling language (IFML)[1] and the WebRatio[2] platform are used for the implementation supporting a model-driven and data-centric development.

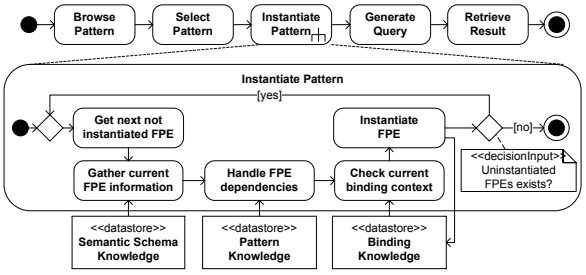

Figure 1: Pattern instantiation guidance activities

The remainder of this paper is organized as follows. Section 2 discusses the semOLAP approach. Section 3 details the semOLAP knowledge graph. Section 4 explains the determination of the navigation flow and the recommendation of possible bindings. Section 5 exemplifies the instantiation of a semOLAP pattern. Section 6 reviews related work. The paper concludes with a summary and an outlook on future work.

2 SEMANTIC OLAP PATTERNS

The notion of patterns is introduced by Alexander et al. [1] where patterns describe how a specific problem in a specific context can be solved while considering existing constraints. In OLAP, an unsatisfied information need represents the problem whereas the specific analysis situation represents the context [10]. A semOLAP pattern can, therefore, be seen as an instruction on how to compose an OLAP query that satisfies the information need in a specific analysis situation. The identification of such patterns is based on the detection of recurring OLAP queries, which are usually abstracted to domain-dependent templates for OLAP reports. To obtain domain-independent semOLAP patterns, such templates are grouped and abstracted (see Fig. 2).

As of now, the identified patterns can be grouped into *basic patterns* and *comparative patterns*. The group of basic patterns

[1]http://www.omg.org/ifml/
[2]http://www.webratio.com

Example	Abstraction	Realization
Compare measures of a subset with its base set. (homogeneous set-base comparison pattern)	Pattern	Pattern Definition
Compare the milk yield of all **<X>** cows with the milk yield of all cows of the farm site **<Y>** per date.	Template	Partial Pattern Instance
Compare the milk yield of all **young** cows with the milk yield of all cows of the farm site **Kremesberg** per date.	Query	Full Pattern Instance

Figure 2: Query abstraction levels

covers generalized multidimensional queries aggregating business events (facts) according to spatial, temporal, and/or semantic aspects. Domain experts perform such a query by joining one fact class with its dimensions, restricting the result of the join using selection criteria (business terms), grouping the result using grouping criteria, and aggregating measures by applying predefined aggregation functions (calculated measures).

In contrast to basic patterns, which are only based on one set, comparative patterns serve to compare two sets. Therefore, a *set of interest* (SI) and a *set of comparison* (SC) need to be defined. The SI is used to specify the primarily focused data which is compared to another set, the SC. For each of these two sets, either the same or different fact class(es), selection criteria, dimension(s), grouping criteria, and/or measure(s) are defined. Depending on the number of shared pattern elements, different types of comparative patterns can be identified. The *homogeneous set-base comparison pattern*, for example, covers all OLAP queries where a subset (SI) is compared with its base set (SC). It is a homogeneous comparison because both SI and SC refer to the same fact class. The grouping criteria, measures, and selection criteria are shared, with the exception of additional selection criteria which are exclusively used to define the SI. The *heterogeneous independent set comparison pattern*, contrary to the previously described patterns, is not restricted to one fact class. It is heterogeneous since two different fact classes are used to define SI and SC. Furthermore, no pattern elements at all must be shared. This also applies to the measures to be compared since they can be based on completely different aggregation functions. The measures from SI and SC can be used to calculate ratios, rates, percentages, proportions, and other complex values.

The definition of such semOLAP patterns is based on semantic web technologies, i.e., RDF, yielding formalized and machine-readable representations. Furthermore, RDF allows to define shared conceptualizations representing calculated measures and business terms (predicates) which can be used during pattern instantiation and linked to domain ontologies. Each pattern definition comprises a textual description, a target language-dependent pattern expression, the pattern result, and the FPEs defining its structure.

As the target audience are domain experts the textual description includes all relevant information needed to instantiate the pattern. Therefore, each semOLAP pattern definition covers a concise pattern name, a description of the analysis situation where it can be applied in, the solution describing the instructions to follow, and an example. In addition to the textual description, a pattern definition contains a pattern expression. This pattern expression is a representation of the query to be generated in a specific target language, e.g., SQL. This representation is enriched by grammar expressions which indicate where certain FPE values must be placed in order to generate an executable query. The result of a pattern is specified by defining which FPEs are returned, i.e., which measures and grouping criteria are returned and how they can be enriched by prefixes to foster differentiation of set-specific elements. It is specified only once in the pattern

definition and not changed during the instantiation. Reusability is fostered, since each result yields a new cube which again can be used as the fact class in other pattern instances.

The structure of an OLAP query is represented by FPEs, which are defined as objects in the pattern definition but treated as properties during the instantiation. The FPE *siFactClass*, for example, is used to define the fact class of the SI during the pattern instantiation of the *heterogeneous independent set comparison pattern*. To support such a behaviour an FPE consists of an element range, a multiplicity, and is part of zero or more pattern element sets (see Fig. 3). The FPE range defines the (sub)type of the values which can be specified during the instantiation. For the FPE *siFactClass*, for example, the range is set to the *Fact* type. Depending on the FPE, the range can be set to (sub)types representing measures, dimensions, dimension attributes, and predicates. The multiplicity, as the name suggests, determines the number of values that can be provided for an FPE during the pattern instantiation, such as *One* or *OneOrMore*. The FPE *siFactClass*, for example, is defined with the multiplicity *One* specifying that only one value of the *Fact* type can be used for the definition of the SI. As already indicated by the prefix of the name *siFactClass*, FPEs can be assigned to pattern element sets, e.g., SC or SI, using the *partOfSet* property. This is especially important during the instantiation of SI and SC, since different selection criteria can be applied to different pattern element sets.

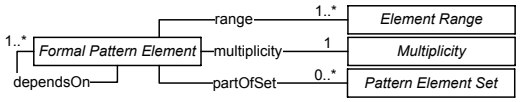

Figure 3: Formal pattern element structure

FPEs can also be related to each other using the *dependsOn* property. The fact class, for example, which stores occurrences of a business event, is the core element of the multidimensional model. These stored occurrences are quantified by measures, hence, there exists a dependency between a measure and its corresponding fact class. Further dependencies exist, since a fact class can be aggregated to different levels of granularities according to its corresponding dimensions and hierarchies. Each fact class has predefined dimensions and each dimension can be assigned also to different fact classes. Dimensions support the aggregation of fact classes to different levels of granularity and, therefore, each dimension has one or more dimension hierarchies which, again, consist of dimension attributes. All these dependencies between FPEs are expressed by *dependsOn* relationships. In addition to the dependencies within the pattern element sets SI and SC, dependencies of FPEs located outside of the pattern element sets can exist. Comparative measures, for example, are defined by using measures from both SI and SC. Comparing two sets requires the specification of FPEs respectively attributes over which those sets can be joined. The join condition can be implicit, if both sets share attributes, or explicitly specified, if no attributes are shared.

The relationships between pattern element sets and FPEs as well as the dependencies between the FPEs themselves yield a graph representation. Fig. 5 depicts such a graph for an instance of the heterogeneous independent set comparison pattern. The *dependsOn* relationships are displayed as grey edges since they are only available in the pattern definition and not directly in the displayed pattern instance (binding graph). Both pattern element sets share the same internal structure, since each of them consists of a *FactClass* and one or more *Measure*, *Dimension*, *DimensionAttribute*, and *Slice* values. The outer FPEs, also called non-set FPEs, *factCorrelation* and *compMeasure* refer to FPEs within the sets. The join condition *factCorrelation* determines which attributes are used to combine the sets whereas the *compMeasure* defines the comparative measure to be calculated.

3 SEMOLAP KNOWLEDGE GRAPH

Guiding users through semOLAP pattern instantiation requires the consideration of the available *semOLAP knowledge graph*, which comprises the three knowledge graphs representing the pattern definition, the semantic schema elements, and current binding of FPEs within the instantiation process (see detailed activity in Fig. 1). Thereby, the semOLAP knowledge graph allows to identify interconnections between a pattern instance and existing values, types, and the underlying schema (see Fig. 4).

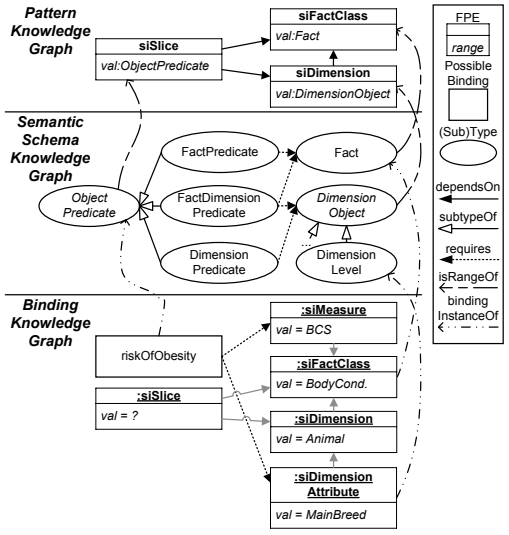

Figure 4: Exemplified semOLAP knowledge graph

A typical OLAP query is composed of the fact class representing the data of interest, grouping criteria, and selection criteria representing logical restrictions regarding temporal, spatial, and semantic aspects. The semOLAP pattern definitions reflect this structure by specifying FPEs and the relationships between them, e.g., the set of selectable measures and dimensions depends on the previously selected fact class. The pattern definition also includes constraints for each FPE: multiplicity, element range, and the pattern element sets to which the FPE is related. This available knowledge, also called *pattern knowledge*, can be exploited during pattern instantiation, e.g., to determine the FPE instantiation order or the type of possible values for an FPE. The pattern knowledge graph in Fig. 4 is an extract of the FPE's *dependsOn* relationships of an SI definition. The *siSlice* depends on the *siFactClass* and the *siDimension*, whereas the *siDimension* depends only on the *siFactClass*. The ranges for these FPEs are represented by the (sub)types of their values, e.g., for the *siFactClass* the FPE range is the type *Fact*.

The types of the FPE ranges are part of the underlying *semantic schema knowledge*. The schema is based on the Dimensional Fact Model (DFM) [8] which allows to conceptually represent multidimensional elements such as fact classes, attributes, dimensions, dimension hierarchies, and relationships between them.

The modeled elements are represented using the RDF Data Cube (QB) [5] vocabulary and its extension QB4OLAP [7], thus creating a semantic multidimensional schema. This RDF representation facilitates the definition of predicates (*ObjectPredicates*) representing business terms as well as calculated measures (*CalculatedMeasures*) which can exceed simple aggregations. The semantic schema knowledge graph in Fig 4 shows the types and subtypes of the range of the FPEs and the structural relationships (dotted directed *requires* edge) between these (sub)types. During the instantiation process this RDF knowledge provides information about the structure of the type, e.g., the type *ObjectPredicate* and some of its subtypes require the structure provided by (sub)types of the ranges of *siFactClass* and/or *siDimension*.

In addition to the pattern and the semantic schema knowledge, the *binding knowledge* has to be considered. It represents the current instantiation, i.e., the bindings of FPEs within the instantiation process. The pattern instance, again represented in RDF, is updated during the instantiation process. The binding knowledge contains the already instantiated FPEs with their values and all currently uninstantiated FPEs. During the binding recommendation process, the binding knowledge needs to be considered, since it reflects the available structure of existing values on the basis of which suitable values can be determined. The current binding knowledge graph in Fig 4 depicts the available fact class value *BCS* and the dimension level value *MainBreed* for *siFactClass* and *siDimension*. These values must be considered to recommend values for *siSlice*, e.g., in order to recommend the *FactDimensionPredicate* value *riskOfObesity*, it is checked if its structurally required values *BCS* and *MainBreed* exist in the binding knowledge.

A guidance approach for query instantiation requires to consider the whole knowledge graph in order to provide navigation and recommendation and to avoid the creation of invalid queries. Valid queries can be only ensured when all relationships between the pattern to be instantiated, the semantic schema elements, and the already provided values are considered.

4 EXPLOITING SCHEMA AND PATTERN KNOWLEDGE FOR INCREMENTAL PATTERN INSTANTIATION

The semOLAP patterns provide a conceptual foundation to compose ad hoc OLAP queries without further assistance. A domain expert, however, requires visual assistance to fulfil this task. They should be enabled to browse existing semOLAP patterns, select the one which fits their information need, and instantiate the semOLAP pattern in order to generate the desired query. Especially the pattern instantiation is a non trivial task since the available knowledge graph, which can be used to determine and restrict possible values for FPEs, must be considered (see Fig. 1).

The guidance process based on semOLAP patterns requires the consideration of the semOLAP knowledge graph as well as an interactive instantiation interface. The interface implementation is based on IFML which supports a data-driven application development following a strict separation of the data model, the hypertext model, and the presentation model [4]. We focus on the hypertext and presentation model since these are crucial for the user interaction. Furthermore, interfaces are generated for the browsing, selection, instantiation, and result retrieval step. To detail the guidance approach and the implementation, the instantiation of the heterogeneous independent set comparison pattern is exemplified (see Fig. 5).

4.1 Navigation Flows

Adapting the idea of logical stratification [12, p. 131-136], we determine a *default navigation flow* by calculating the corresponding level of each FPE. The calculation of the levels is based on the FPEs from the *pattern knowledge* and simple rules: FPEs with no outgoing *dependsOn* edge are assigned to level 0; FPEs which have one or more outgoing *dependsOn* edges are assigned to the highest level of the referred FPEs plus one; these steps are repeated until the level assignments are not changed any more (see Algorithm. 1).

```
repeat
    forall formalPatternElement fpe in
     patternKnowledgeGraph do
        level[fpe] := 0;
    end
    repeat
        forall formalPatternElement fpe in
         patternKnowledgeGraph do
            forall dependsOn dp in fpe.dependsOn do
                targetFpe := dp.target;
                if level[fpe] <= level[targetFpe] then
                    level[fpe] := level[targetFpe]+1;
                end
            end
        end
    until there are no changes to any level or a level
     exceeds the number of formal pattern elements;
until all levels of abstraction are processed;
```

Algorithm 1: Level computation

An exemplified application of this algorithm is the calculation of the SI levels depicted in Fig. 5. The calculated level assignments are indicated by the number in the left corner of the instantiated FPEs. The first number indicates the assigned level whereas the second number indicates the sequence within the default navigation flow. The *siFactClass* is assigned to level 0 since it has no outgoing *dependsOn* edges; *siMeasure* and *siDimension* are assigned to level 1 due to their dependence on *siFactClass*; *siDimensionAttribute* and the *siSlice* are assigned to level 2 due to their dependence on *siDimension*. This algorithm, however, is not limited to the FPEs in the pattern element sets SI and SC, it can also be applied to the next level of abstraction. Each pattern element set can be also seen as an FPE of an outer graph. Considering this abstraction level both SI and SC represent FPEs without outgoing *dependsOn* edges which are referenced by the FPEs *factCorrelation* or the *compMeasure*.

The default navigation flow can be determined by considering the *dependsOn* relationships between the FPEs and the assigned levels. It starts from FPEs assigned to the lowest level, i.e., from FPEs assigned to level 0. If multiple FPEs are assigned to the same level an arbitrary navigation flow order can be specified for them. These steps are repeated for the next levels until all levels are processed. The result is a default navigation flow linking the interface elements of the FPEs during the instantiation of a selected pattern.

The dotted directed *linksTo* edges in Fig. 5 represent the default navigation flow of the heterogeneous independent set comparison pattern. It starts with the value specification of *siFactClass*, followed by the specification of *siMeasures* and *siDimension*. The order of these last two FPEs is interchangeable since both of them

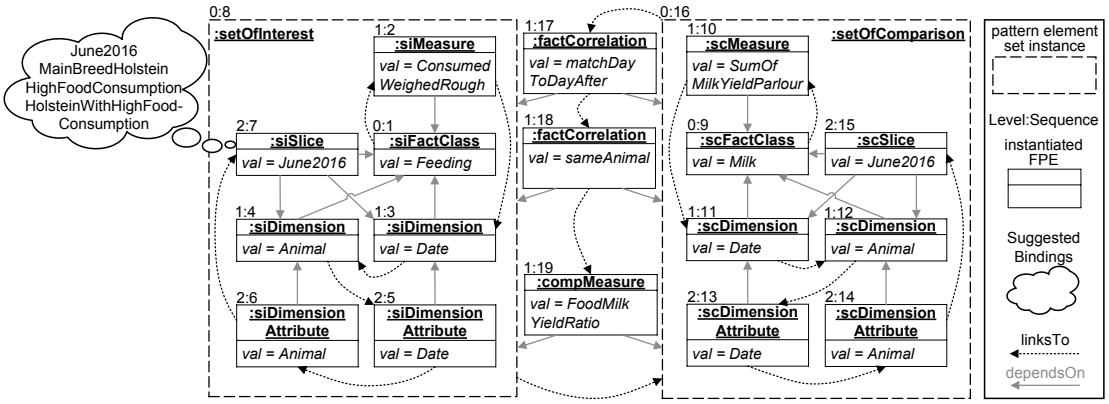

Figure 5: Enriched binding graph of a heterogeneous independent set comparison pattern example

are assigned to level 1. At level 2, values must be specified for the dimension attributes *siDimensionAttribute*, representing the grouping criteria, and the *siSlice*, representing set-specific predicates. The selectable dimension attributes depend on the specified dimensions. Depending on the values specified in the previous levels, only specific types of predicates can be specified for the *siSlice*. In the heterogeneous independent set comparison pattern the FPE *dependsOn* relations of SI and SC are the same, therefore SI's navigation flow can be applied to the SC analogously. To finish the instantiation of the pattern the *factCorrelation* attribute, used for joining, and the *compMeasure*, which determines the type of comparison to be performed, have to be specified.

The default navigation flow only allows slight adaptations, such as changing the FPE order within one level, e.g., either *siMeasure* or *siDimension* can be instantiated first. Additional adaptations, however, have to be supported since a user might not want to start with the specification of the FPE *siFactClass*, instead they might want to start with other FPEs. As discussed in [3], a user knows prior to the query composition which measure(s) they want to retrieve, therefore a user typically starts with the selection of the desired measure(s). This is especially relevant for ad hoc query composition, since a user wants to retrieve something that is not covered by existing reports. Facilitating such a *custom navigation flow* requires the adaptation of the default navigation flow which is based, so far, on the FPEs' *dependsOn* relationships and the assigned levels. The navigation flow must be detached from these *dependsOn* relationships to provide such flexibility. A custom navigation flow cannot be determined automatically, instead it must be specified manually in the course of system configuration. In exchange, the custom navigation flow allows to move arbitrarily between the FPEs, e.g., allowing to navigate from *siMeasure* to *siFactClass*.

4.2 Binding Recommendation

The navigation flow allows to move from one interface element to another while providing values for the corresponding FPEs. The user can be guided in this process by having bindings recommended for the FPE values. Therefore, the range of the current FPE (available in the *semantic schema knowledge*), the *dependsOn* relationships between FPEs (*pattern knowledge*), and the bindings of other FPEs (*binding knowledge*) need to be considered. To illustrate this, we exemplify the instantiation of the FPE *siSlice* in the heterogeneous independent set comparison pattern by recommending bindings of the *FactDimensionPredicate* subtype; this approach can be applied to all other FPEs and (sub)types as well.

Each FPE specified in the pattern definition is represented as a property during the pattern instantiation. The range of each FPE determines the type of possible bindings, e.g., the range of *siSlice* is *ObjectPredicate*. Due to the complexity of the multidimensional model, each range can cover multiple subtypes which are a part of the *semantic schema knowledge*, e.g., *ObjectPredicate* is a subtype of *Predicate* and a supertype of *FactPredicate*, *DimensionPredicate*, and *FactDimensionPredicate*. Consequently it is possible to select bindings of the types *FactPredicate*, *DimensionPredicate*, or *FactDimensionPredicate* for the FPE *siSlice* (see pattern knowledge and the semantic schema knowledge graph in Fig. 4). We focus here on bindings of the type *FactDimensionPredicate*.

Recommending bindings for the current FPE requires to retrieve all other FPEs that the current FPE depends on – the *depending FPEs*. To this end, *dependsOn* relationships from the *pattern knowledge graph* are used. Considering, for example, the *dependsOn* relationships of the *siSlice* allows to identify the depending FPEs *siFactClass* and *siDimension*. For each subtype of the current FPE's range, each depending FPE is processed separately. For the sake of simplicity we refer to the (sub)type of the current FPE's range as *current subtype* and to the subtypes of the depending FPEs' range as *depending subtypes*. For example, for the current subtype *FactDimensionPredicate*, as a subclass of *siSlice's* range, the depending FPE *siDimension* is processed. Therefore, the (sub)types of the depending FPE's range are retrieved. For the range of the depending FPE *siDimension*, for example, the subtypes are *DimensionLevel* and *DimensionRole*[3].

For the current subtype *FactDimensionPredicate* and the depending subtypes *DimensionLevel* and *DimensionRole* the possible basic relations *FactDimensionPredicateRelatesToDimensionLevel* and *FactDimensionPredicateRelatesToDimensionRole* need to be considered. A *basic relation* is used to represent the structural relationship of a current subtype to a depending subtype (see *requires* relationships in the semantic schema knowledge graph in Fig. 4). Not all possible basic relations derived from current and depending subtypes actually exist. The *siSlice*, for example, depends on *siDimension* but the *FactPredicate*, which is a subtype of *siSlice's* range, does not have a basic relation to either subtypes of *siDimension's* range *DimensionLevel* nor *DimensionRole*. Therefore, only the actually existing basic relations are then used

[3] A dimension role is used to reference dimensions using different names, e.g., the dimension animal can be references using the dimension role dam animal.

to determine potential bindings of the current subtype for the current FPE. Each of the existing basic relations is represented by a predefined SPARQL Protocol And RDF Query Language (SPARQL) query which checks all available values of the current subtype in order to determine potential bindings.

The bindings of the depending FPE that are of the depending subtype are checked against the required structure of each available value of the current subtype. The structure of values is represented by relationships in the multidimensional schema, predicates, and measures, e.g., the *FactDimensionPredicate riskOfObesity requires* the dimension attribute *MainBreed* and the measure *BCS* (see Fig. 4). If the required structure regarding the current basic relation is available in the structure of the depending FPE binding, the value of the current subtype is added to the list of potential bindings of the current depending FPE. Determining if *riskOfObesity*, for example, is a potential binding for the current FPE, the underlying SPARQL query of the basic relation *FactDimensionPredicateRelatesToDimensionLevel* checks whether a binding of the depending subtype *DimensionLevel* (of the depending FPE *siDimension*) exists which contains the dimension attribute *MainBreed*. Since available values of the current subtype are checked against multiple bindings of different depending subtypes, the list of potential bindings of the current depending FPE is extended continuously, e.g., bindings of *siDimension* are either of the depending subtype *DimensionLevel* or *DimensionRole*. This concludes the calculation of potential bindings for the first depending FPE *siDimension* for the current subtype *FactDimensionPredicate*.

The current subtype, however, might require depending subtypes of more than one depending FPE, e.g., *FactDimensionPredicate* requires the depending subtype *Fact* of *siFactClass* and *at least one* of the depending subtypes of *DimensionObject* of *siDimension* (see semantic schema knowledge graph in Fig. 4). Therefore, the potential bindings of these depending FPEs have to be determined which results in a list of potential bindings for each depending FPE. Only the potential bindings present in all these lists are possible bindings which can be recommended as a binding for the current FPE, e.g., *riskOfObesity* requires a binding of subtype *DimensionLevel* or *DimensionRole* which contains the dimension attribute *MainBreed* as well as a binding of subtype *Fact* which contains the measure *BCS*.

The calculation of possible bindings is repeated for all other current subtypes, i.e., *FactPredicate* and *DimensionPredicate*. Finally, all possible bindings of all subtypes of the current FPE's range are combined and returned to the user as binding recommendations for the FPE currently being instantiated.

Considering the relationships between a current subtype and its depending subtypes using the corresponding basic relations enables the recommendation of bindings along the default navigation flow. These corresponding basic relations are following the *dependsOn* relationships between the FPEs. Supporting recommendations for the custom navigation flow, however, requires the extension of the *dependsOn* relationships to bidirectional ones. Up to now, the FPE's *dependsOn* relationships have reflected a hierarchical structure, i.e., *siFactClass* serves as the root whereas all other FPEs are either directly or indirectly depending on it. This hierarchical view yields a directed graph which can be traversed from top to bottom, i.e., the default navigation flow. Representing the *dependsOn* relationships bidirectionally leads to a non-hierarchical view which can be traversed beginning from any FPE. No new relationships between FPEs are introduced, only existing unidirectional *dependsOn* relationships are extended to bidirectional ones. After the *dependsOn* relationships are extended, the corresponding basic relations have to be defined. Therefore, the basic relations of the subtypes of the FPE's ranges need to be extended by relations in the opposite direction, e.g., for the FPEs *siSlice* and *siFactClass* with their corresponding ranges *ObjectPredicate* and *Fact* the existing basic relations are extended by *FactRelatesToFactDimensionPredicate*, *FactRelatesToFactPredicate*, and *FactRelatesToDimensionPredicate*.

If a user, for example, starts to select a binding for *siMeasure* and then navigates to the FPE *siFactClass*, the basic relation *FactRelatesToObjectCalculatedMeasureRelates* can be used to determine *Fact* values, which provide the necessary structure for the previously specified *siMeasure* value(s). This, again, takes the semOLAP knowledge graph into account. The binding recommendations for a custom navigation flow, however, faces also limitations. For example, the instantiation could start with the specification of the *siMeasure* value, followed by the *siDimension* value and continuing with the *siFactClass* value. Recommending possible bindings for *siDimension* would not be possible, since no basic relations between the subtypes of *siMeasure* and *siDimension* ranges exist in the heterogeneous independent set comparison pattern. All potential bindings, in this case all values of the subtypes *DimensionRole* and *DimensionLevel*, would be recommended as possible bindings. This issue can be solved by introducing new basic relations between the subtypes of *siMeasure* and *siDimension*. These new basic relations, which do not follow existing *dependsOn* relationships, can be created by considering existing basic relations between subtypes of *siMeasure* and *siFactClass* and subtypes of *siFactClass* and *siDimension*. In contrast to *siDimension*, possible bindings could be recommended for the *siFactClass* without new basic relations, since there exist *dependsOn* relationships between the *siFactClass* and both *siDimension* and *siMeasure*. The bidirectional basic relations *FactRelatesToDimensionRole*, *FactRelatesToDimensionLevel*, and *FactRelatesToObjectCalculatedMeasure* are therefore used. Considering these relations allows to recommend bindings for the instantiation of the FPE *siFactClass*, however, it could be possible that no bindings at all could be recommended. This would be the case if, for example, a combination of values for *siMeasure* and *siDimension* is selected which cannot be structurally supported by any value of *siFactClass*. To resolve this issue the user would need to navigate back and edit the corresponding FPE values, otherwise the instantiation could not be continued. The default navigation flow is not affected by these limitations at all, since it considers the logical dependencies derived from the multidimensional model.

Binding recommendations are restricted by the availability of existing bindings of the depending FPEs. The advantage of this approach is that the basic relations are defined only once and can be reused multiple times since FPEs with the same dependencies are used within different pattern definitions. For example, the basic relation between the subtype *FactDimensionPredicate* and *DimensionLevel* occurs in the basic multi-aggregation patterns as well as in other comparison patterns such as the homogeneous set-base comparison pattern. New FPEs with currently not considered *dependsOn* relationships can be introduced as part of new semOLAP patterns. To handle these dependencies only their basic relations and the underlying SPARQL queries need to be defined once. Contrary to this case, new semOLAP patterns using only considered *dependsOn* relationships, can be instantiated without further effort.

5 EXEMPLIFIED PATTERN INSTANTIATION

The guidance approach is exemplified by instantiating the heterogeneous independent set comparison pattern following the default navigation flow. Even though custom navigation flows could be supported, only the unidirectional basic relations are implemented so far. To illustrate our approach we consider a domain expert who wants to compose an ad hoc query which calculates the ratio of the consumed food of one day (SI) and the milk yield of the next day (SC) for the same animal in June 2016 per date and animal (see FPE bindings in Fig. 5). The resulting ratio is used by the expert to see whether the amount of food fed the day before impacts the milk yield of the next day. Therefore, a data cube containing the two fact classes *Milk* and *Feeding* with the shared dimensions *Date*, *Farm Site*, and *Animal* is accessed (see DFM model in Fig. 6).

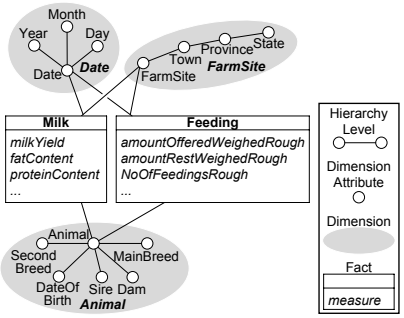

Figure 6: DFM of facts of interest

The instantiation starts with the specification of the SI and its corresponding FPEs. The available fact classes *Milk* and *Feeding* are recommended, besides others, to the domain expert. The domain expert selects *Feeding* as the binding value for the *siFactClass* FPE. The specified fact class *Feeding* and the corresponding basic relations are used to determine possible bindings for *siDimension*, i.e., the dimensions *Animal*, *Date*, and *FarmSite*. The domain expert selects the *Animal* and *Date* dimensions as values for *siDimension* and continues with the specification of the measure of interest *siMeasure*, i.e., the *consumedWeighedRough* representing the summarized values of consumed weighed roughage. For the *siDimensionAttribute* the dimension attributes *Animal* and *Date* binding values are selected. The names of the values of *siDimension* and *siDimensionAttribute* are the same, even though they represent different FPEs. This is the result of naming conventions, since the dimension's name is used as the name of the identifying dimension attribute (see Fig. 6). Based on the dimension values *Animal* and *Date* and the fact value *Feeding*, *DimensionPredicate* values, such as *June2016* or *MainBreedHolstein*, *FactPredicate* values, such as *HighFoodConsumption*, and *FactDimensionPredicate* values, such as *HolsteinWithHighFoodConsumption*, are recommended to the domain expert as possible values for *siSlice* (see recommended bindings in Fig. 5). To finish the instantiation of the SI, the domain expert selects *June2016* as the binding for *siSlice*.

The instantiation of SC starts after the instantiation of SI is finished. Since the default navigation graph of SC and SI are isomorph, the domain expert is guided through the same steps. The domain expert provides the same value bindings to *scDimension*, *scDimensionAttribute*, and *scSlice* already used for the analogous FPEs of the SI. This is possible since in this query SI and SC share the dimensions, dimension attributes, and slices. Only for the FPEs *scFactClass* and *scMeasure* different bindings have to be specified. The *Milk* fact is specified as the binding of *scFactClass* and *SumOfMilkYieldParlour* measure as the binding of *scMeasure*.

After both sets are instantiated, the domain expert specifies the bindings *matchDayToDayAfter* and *sameAnimal* for the *factCorrelation* and *FoodMilkyieldRatio* for the *compMeasure* to finish the instantiation process. Similar to the previous instantiation steps, other possible values for these two FPEs are provided, however, only the mentioned are relevant for the intended query. The *factCorrelation* value *matchDayToDayAfter* combines the fact occurrences of SI of one specific day with the fact occurrences of SC of the next day whereas *sameAnimal* restricts the combinations to the same animals. The *FoodMilkyieldRatio* calculates the ratio between SI's *consumedWeighedRough* and SC's *SumOfMilkYieldParlour*. After all FPE values are specified, a summary of the pattern instance is provided (see Fig. 7). This summary provides an overview of all FPEs of a semOLAP pattern instance and indicates which FPEs are differently specified in SI and SC. The differences are color-encoded to ease their identification. The structure of the overview is independent of the semOLAP pattern but it can be adapted by the developer to consider characteristics of certain patterns.

SetOfInterest		SetOfComparison	
PatternElement	Instance	PatternElement	Instance
siFactClass	Feeding	scFactClass	Milk
siDimension	Animal	scDimension	Animal

Figure 7: Detail of the pattern instance summary

Finally, to satisfy the domain expert's information need, the OLAP query is generated and sent to the underlying ROLAP system. The result of this OLAP query is visualized and can thereby be interpreted by the domain expert. The domain expert can reuse this pattern instance for future analysis situations and adapt it to fit their information need. Besides fully-instantiated patterns, partial instances can be specified as well, e.g., the fact, dimension, dimension attributes, and measures are predefined and only additional selection criteria can be specified. These partially-instantiated patterns can be reused as domain-dependent query templates. A video[4] of this instantiation process is provided to demonstrate the current state of the implementation.

6 RELATED WORK

The guidance of users during the OLAP query composition is mostly accompanied by providing suitable visualizations of the query elements. The Semantic Data Warehouse Model (SDWM [3]) allows for a visual specification of multidimensional queries. The SDWM does not represent semantic representations of the multidimensional data model, e.g., using QB and QB4OLAP, instead the SDWM considers both the operational requirements as well as the semantics of the business processes to be modeled [3]. Therefore, templates which are based on the SDWM are provided to users, serving as configurable reports [3]. Each of these templates consists of predefined measures, dimensions, and dimension attributes which are related to each other. The relationships between these template elements are visualized to represent the dependencies between the measures and dimensions. The user specifies the OLAP query by either adding/removing new measures or by selecting the dimension hierarchy levels. Similar to our approach, only possible dimension and dimension

[4] https://www.youtube.com/watch?v=BLt6heO7WKY

attribute values are provided to the user, however, this is not ensured through reusable basic relations. Furthermore, additivity checks are performed, which restrict aggregations to only possible measures, e.g., it does not make sense to summarize the food to milk ratio over time. The proposed approach using the SDWM, however, does not provide the abstraction of semOLAP pattern, since it focuses on case-specific templates which are restricted to a corresponding fact class. The user is not able to specify fact values nor complex predicate values; only simple restrictions are supported. Furthermore, the composition of ad hoc queries targeting multiple fact classes is not considered at all.

Another query visualization interface is Polaris [11] which led to the development of Tableau[5]. Polaris focuses on analyzing, querying, and visualizing multidimensional relational databases although newer versions of Tableau support other data structures as well. Instead of focusing on ad hoc queries, it primarily supports the explorative data analysis approach by providing an interactive visualization of both the query and the result. The query is defined by a visual specification within a table-based interface, which allows to specify dimensions, measures, and grouping and filter criteria along with possible visualisation options. Corresponding queries are generated using an underlying table algebra. Ad hoc queries can be formulated since all functionalities for composition are provided, however, analysis situation-specific guidance is not supported. The user is not guided while creating, for example, comparisons of sets from one or multiple fact classes, even though, the necessary functionality is available. The application of filters is provided, however, these are restricted to simple expressions; predicates representing business terms are not supported. Furthermore, calculated measures relate only to the fact class where they where specified, whereas calculated measures and object predicates in semOLAP are independent of the fact class as long as the necessary structure is provided by any target fact class. This is possible since our approach is not directly based on the relational data model, unlike Polaris [11], instead it is based on the multidimensional data model. In comparison to the Polaris approach, we support reusing and editing instantiated semOLAP queries to match the new information demand. SemOLAP queries can, additionally, be used as the data input for other semOLAP queries since each result is representing a possible fact class as it consists of measures and dimension attributes.

7 SUMMARY AND FUTURE WORK

We have proposed a guided query composition approach based on semOLAP patterns. The semOLAP pattern approach provides the conceptual foundation to allow ad hoc query composition by domain experts. This conceptual foundation is realized by a data-centric and model-driven implementation which guides the domain expert during the instantiation of the FPEs. For each FPE instantiation, bindings are recommended by considering the semOLAP knowledge graph which comprises knowledge about the semantic schema, the pattern structure, and the current FPE binding. Therefore, basic relations are introduced which are used to check structural dependencies between the (sub)types of the FPE ranges. This allows users to move through the FPE instantiation process and select recommended bindings which consider the current instantiation state, the FPE's type information, and constraints of the FPE itself.

For each semOLAP pattern a default navigation flow is calculated and provided to guide the user through the instantiation process. If an instantiation sequence other than suggested by the default navigation flow is more convenient for a particular pattern, a custom navigation flow can be configured by a developer. To this end, basic relations are extended to bidirectional relations, allowing to recommend bindings independent of the navigation sequence. Even navigation flows between FPEs which are not represented in the FPE's *dependsOn* relationships can be supported, hence, leading to a maximum level of flexibility. The drawback of this flexibility is that instantiation situations can occur where no bindings at all can be recommended, since an unsupported FPE value combination is selected.

Future work will include displaying available information currently hidden in the semantic representation of schema elements, e.g., the measure *SumOfMilkYieldParlour* is linked to the AGROVOC [6] ontology which includes a concise definition of the measure. The calculation of a navigation flow can also consider the number of available FPE values. This would require the consideration of the selectivity of FPE values, i.e., it is preferable to start with the FPE value specification which has the highest potential to reduce the number of potential values of other FPEs. Furthermore, the visualization of the result, which is currently limited to a table representation, will be extended. In the future domain- and data-dependent visualizations will be automatically applied to the result. This visualization will also comprise a generated caption as well as a generated result description.

REFERENCES

[1] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Joaquim Romaguera i Ramió, Max Jacobson, and Ingrid Fiksdahl-King. 1977. *A pattern language.* Oxford University Press.

[2] Jeffrey Bewley. 2010. Precision dairy farming: advanced analysis solutions for future profitability. In *Proceedings of the first North American conference on precision dairy management, Toronto, Canada.* 2–5.

[3] Michael Böhnlein, Achim Ulbrich-vom Ende, and Markus Plaha. 2002. Visual Specification of Multidimensional Queries based on a Semantic Data Model. In *Vom Data Warehouse zum Corporate Knowledge Center.* Springer, 379–397.

[4] Marco Brambilla and Piero Fraternali. 2014. *Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML.* Morgan Kaufmann.

[5] Richard Cyganiak and Dave Reynolds. 2014. *The RDF Data Cube Vocabulary.* W3C Recommendation. W3C. http://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/.

[6] Wayne W. Eckerson. 2008. Pervasive business intelligence: Techniques and technologies to deploy BI on an enterprise scale. *TDWI Best Practices Report* (2008).

[7] Lorena Etcheverry and Alejandro A. Vaisman. 2012. QB4OLAP: A New Vocabulary for OLAP Cubes on the Semantic Web. In *Proceedings of the Third International Conference on Consuming Linked Data.* 27–38.

[8] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. 1998. The dimensional fact model: A conceptual model for data warehouses. *International Journal of Cooperative Information Systems* 7, 2-3 (1998), 215–247.

[9] Thomas Neuböck, Bernd Neumayr, Michael Schrefl, and Christoph Schütz. 2014. Ontology-Driven Business Intelligence for Comparative Data Analysis. In *eBISS 2013.* LNBIP, Vol. 172. Springer, 77–120.

[10] Christoph G. Schuetz, Simon Schausberger, Ilko Kovacic, and Michael Schrefl. 2017. Semantic OLAP Patterns: Elements of Reusable Business Analytics. In *OTM 2017 (LNCS)*, Vol. 10573. Springer.

[11] Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics* 8, 1 (2002), 52–65.

[12] Jeffrey D. Ullman. 1988. *Principles of Database and Knowledge-Base Systems, Volume I.* Principles of computer science series, Vol. 14. Computer Science Press.

[13] Martin Wischenbart, Dana Tomic, Michael Iwersen, Michael Schrefl, and Valentin Sturm. 2017. agriProKnow – Prozessbezogenes Informationsmanagement in Precision Dairy Farming. In *Proceedings der 13. Tagung Bau, Technik und Umwelt in der landwirtschaftlichen Nutztierhaltung (BTU-Tagung 2017).*

[5] https://www.tableau.com

[6] http://aims.fao.org/vest-registry/vocabularies/agrovoc-multilingual-agricultural%2Dthesaurus