A Model-Driven Heuristic Approach for Detecting Multidimensional Facts in Relational Data Sources

Andrea Carmè[1], Jose-Norberto Mazón[2], and Stefano Rizzi[3]

[1] Iconsulting, Italy
a.carme@iconsulting.biz
[2] Lucentia Research Group
Dept. of Software and Computing Systems
University of Alicante, Spain
jnmazon@dlsi.ua.es
[3] DEIS - University of Bologna, Italy
stefano.rizzi@unibo.it

Abstract. Facts are multidimensional concepts of primary interests for knowledge workers because they are related to events occurring dynamically in an organization. Normally, these concepts are modeled in operational data sources as tables. Thus, one of the main steps in conceptual design of a data warehouse is to detect the tables that model facts. However, this task may require a high level of expertise in the application domain, and is often tedious and time-consuming for designers. To overcome these problems, a comprehensive model-driven approach is presented in this paper to support designers in: (1) obtaining a CWM model of business-related relational tables, (2) determining which elements of this model can be considered as facts, and (3) deriving their counterparts in a multidimensional schema. Several heuristics –based on structural information derived from data sources– have been defined to this end and included in a set of Query/View/Transformation model transformations.

1 Introduction

The development of data warehouses is based on detecting multidimensional elements from a detailed analysis of data sources. Among multidimensional elements, facts are those of highest importance since they represent events of interests for knowledge workers. Therefore, several techniques, such as guidelines or glossaries, have been developed so far to support designers in detecting multidimensional roles of elements in a relational schema (including facts). For example, in a retail domain, a table called Sales is likely to cover the role of a fact. However, these techniques may become tedious and time-consuming when the application domain is complex (in a medical domain, is a table called FertilityCycle a fact?) or, even worse, when table names are meaningless (what is the multidimensional counterpart of a table called SP_CCCM?).

Other approaches arose to support designers in tackling this task in a more automated manner [1,2,3]. However, these are focused on automatically detecting other multidimensional concepts (such as dimension hierarchies) rather than facts, so discovering facts still relies on informal techniques. Furthermore, most approaches assume that data sources are well-documented or documentation can be easily obtained; unfortunately, this is not generally true [4], and even if some documentation exists, it is likely to be out-of-date with respect to the actual data sources.

To overcome these drawbacks, in this paper we present an approach for formalizing fact detection from relational data sources without requiring additional documentation. Our approach is based on a set of heuristics, elicited from some real-world case studies we are working on. These heuristics use some syntactical information derived from the data sources, thus guiding designers in the detection of facts independently of their knowledge about the application domain. We have formalized these heuristics by means of QVT (Query/View/Transformation) transformations in a model-driven perspective, in such a way that the final multidimensional schemata are derived with a high degree of automation, thus saving time and costs. Basically, our approach consists of three tasks (see Fig. 1): (1) detect clusters of business-related tables within data sources and derive their relational CWM model, (2) support designers in properly determining which elements of this model can be considered as facts by means of a set of heuristics-based QVT model transformations, and (3) model facts, together with their dimensions and measures, in a multidimensional schema.

The remainder of this paper is structured as follows. Section 2 briefly describes the current approaches for discovering multidimensional facts. Section 3 describes our heuristics and the definition of model transformations for detecting facts. Section 4 presents an implementation of our approach and draws the conclusions.

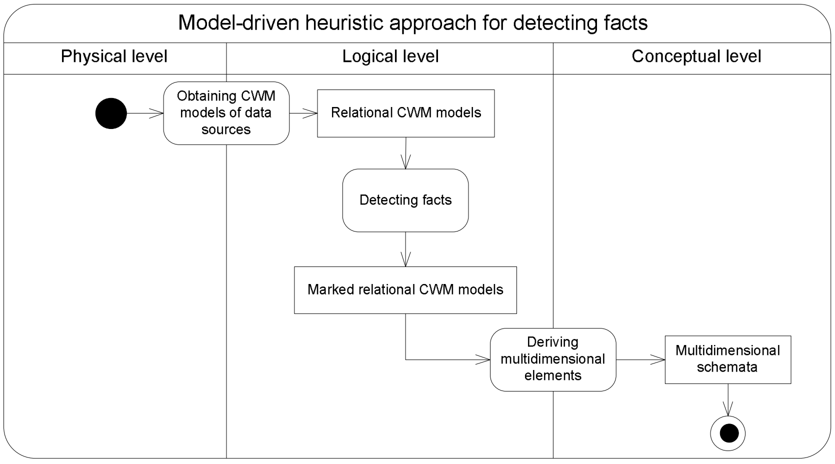

Model-driven heuristic approach for detecting facts
Physical level
Logical level
Conceptual level
Obtaining CWM models of data sources
Relational CWM models
Detecting facts
Marked relational CWM models
Deriving multidimensional elements
Multidimensional schemata

Fig. 1. Overview of our approach for detecting facts

2 Related Work

Most approaches for deriving multidimensional schemata from relational data sources (e.g., [5,6,7,8]) propose informal mechanisms (such as guidelines or glossaries) to support designers. In order to increase the level of automation of this task, other approaches use heuristics to determine which tables are good candidates to become facts. Phipps and Davis [1] propose to consider every entity in an Entity-Relationship schema that contains numerical attributes as a fact, which may be unfeasible since (1) most entities in a schema would be selected, and (2) it is assumed that an up-to-date conceptual schema of data sources is available. Jensen et al. [2] consider not only the presence of measures, but also table cardinality to identify facts; though this approach builds on a reverse-engineering stage in which relational metadata is obtained from data sources, its success highly depends on the skill of domain experts.

Two automated approaches for detecting facts are presented in [3] and [9]. Song et al. [3] propose structural heuristics to detect facts from an Entity-Relationship schema: all entities with a high number of many-to-one relationships are candidates to become facts. Not realistically, they assume that a conceptual schema is always available. Romero and Abelló [9] detect facts by expressing multidimensional SQL queries over relational data sources, and assume that those aggregated attributes in the SELECT clause which are not included in the GROUP BY clause belong to a table that is a potential fact. However, this approach depends on the ability of the users to express their own information requirements as SQL queries.

Our work is inspired by [10], that considers relational data sources as legacy systems whose documentation either is not available, or cannot be obtained, or is too complex to be easily understood through a manual analysis. To overcome these problems, they consider the development of a data warehouse as a modernization scenario which addresses the analysis of the available data sources aimed at discovering multidimensional structures. These structures are then used to derive a data-driven multidimensional schema or reconcile a requirement-driven multidimensional schema with data sources. However, the heuristics for detecting facts presented in that work are rather simplistic and deliver a single solution, which may hide the analysis potential of data sources.

3 Model-Driven Heuristic Approach for Detecting Facts

Our model-driven approach aims to support designers in marking tables from relational data sources as facts. Each table can be differently marked, thus suggesting several possibilities to designers. A set of heuristics for determining which tables are good candidates for being facts, mainly based on an analysis of functional dependencies, have been developed and formalized by using QVT (Query/View/Transformation) [11] model transformations. Our approach assumes that all database constraints (primary and foreign keys) are known, which is perfectly reasonable since these constraints can be nimbly derived [12].

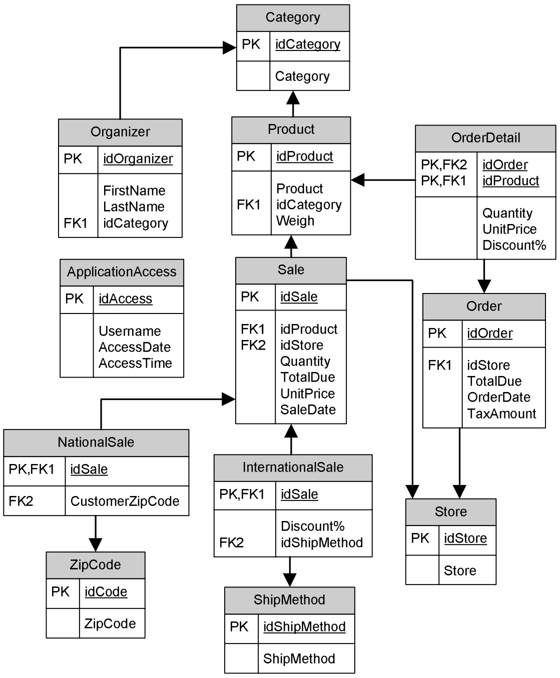


Fig. 2. Relational schema for the running example

The example we will use throughout the paper is based on the retail domain (see Fig. 2) and summarizes situations we have detected in a real case study we are working on at the Spanish fertility institute *TAHE Fertilidad*[1], which we cannot show due to confidentiality issues. Data related to sales and orders are stored, as well as stores, products, etc. Sales are specialized into national and international ones. The OrderDetail relation allows to include several products in each order.

3.1 Obtaining CWM Models of Data Sources

This phase concerns the extraction of relational elements (tables, columns, and constraints) from data sources by querying the DBMS data dictionary. It consists of two steps: (1) delimiting the relational elements related to the application domain, and (2) creating their models based on CWM.

The rationale behind the first step is that, in real-world scenarios, data sources not only store interesting data for analysis but also data about instance feeding applications, security, audit, and so on, that should be ignored when facts are being detected. The benefits of this pre-processing step are twofold: on the one hand, useless elements are not considered; on the other, heuristics will be

[1] `http://www.tahefertilidad.es`

more reliable because the required measures will be calculated by considering only interesting relational elements. Relational elements are first grouped into clusters, using a graph theory algorithm that computes connected graph components [13]. The output is a set of directed, connected graphs whose nodes and edges represent relations and functional dependencies, respectively. Then the designer, in collaboration with domain experts, manually determines which clusters are useful for analysis. In our running example, the cluster containing table ApplicationAccess is not considered, since it is supposed to be unrelated to the business domain.

During the second step, a *relational CWM* (rCWM) model is created for each selected cluster. *Common Warehouse Metamodel* (CWM) [14] consists of a set of metamodels for representing data warehouse and business intelligence metadata, including a relational metamodel that allows relational elements to be easily represented. The next phases of our approach are applied separately to each rCWM model created. Fig. 3 shows part of the rCWM model for our running example.

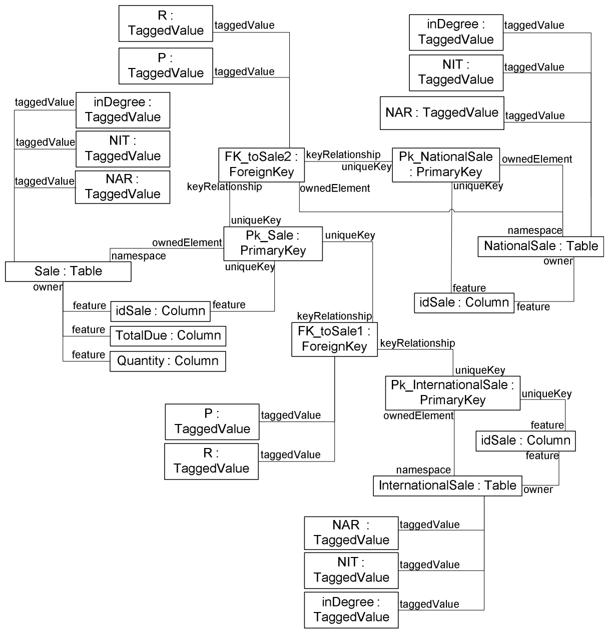


Fig. 3. Part of the relational CWM model for the running example

3.2 Detecting Facts

The fact detection process (Fig. 4) consists of several steps aimed at (1) marking relationship cardinalities, (2) calculating the in-degree of tables, (3) marking facts, (4) marking dimensions and measures, and (5) spawning analysis contexts. Note that several marks can be applied to each relational element, by adding

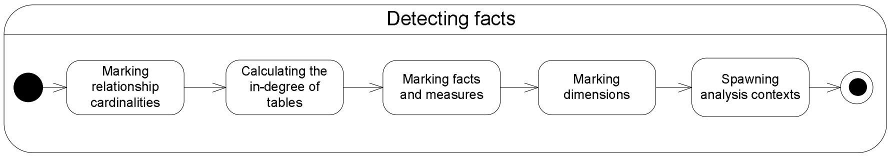

Detecting facts
Marking relationship cardinalities
Calculating the in-degree of tables
Marking facts and measures
Marking dimensions
Spawning analysis contexts

Fig. 4. Fact detection process

values to the *description attributes* provided by CWM. Before explaining the process steps, we describe the heuristics they rely on.

Heuristics. Our heuristics are based on a set of measures calculated from the tables of the rCWM model.

1. The first heuristics states that a table may be a fact if it contains a higher number of instances (NIT) than most other tables. The rationale is that a large table is frequently updated because it stores data related to dynamic events of a business process. The NIT value is retrieved querying data sources through a simple SQL query.
2. The second heuristics states that a table may be a fact if it has a large ratio of numerical attributes: $NAR = NNA/NTA$, where NNA is the number of numeric attributes and NTA is the total number of attributes of a table.
3. The third heuristics states that a table may be a fact if it has a low in-degree, i.e., few or no incoming foreign keys (an incoming foreign key for table T is a foreign key referencing the primary key of T).

To quantify qualitative terms such as "high" and "few", we computed three thresholds. Thresholds for NIT and NAR are calculated using the statistical percentile concept [15]. We have chosen the upper quartile (75-th percentile) as the NIT threshold and the lower quartile (25-th percentile) as the NAR threshold because this gave good results in our case study. Of course, further tests will be needed to find the best percentile to be used in general cases. The in-degree threshold is fixed to 1, which means considering as potential facts only tables with one or no incoming foreign key. We use 1 instead of 0 to consider some specific patterns that we will explain in the following subsections.

Each heuristic measure is stored in a CWM tagged value connected to the related table, as shown in Fig. 3. Thresholds are stored using tagged values linked to the package that contains relational elements.

Marking relationship cardinalities. The relational model has a limited expressiveness. Specifically, one-to-one relationships, that have an ad-hoc representation in the Entity-Relationship model, are not explicitly modeled in a relational schema. Indeed, the existence of a foreign key between two tables does not explain if the relationship between these tables is many-to-one or one-to-one. Since this knowledge is necessary for our approach, we use two transformations to single out two kinds of one-to-one relationships that we will call, respectively, *strong* and *weak*.

– Strong one-to-one relationships are *schema-based* since they are derived and validated within the schema structure. Precisely, a strong one-to-one relationship between two tables T and S is detected when the primary key of T is a foreign key referencing S. A QVT transformation checks this pattern inside rCWM models and marks the foreign keys involved as one-to-one.
– Weak one-to-one relationships are *instance-based*, since they are elicited from data sources instances. A weak one-to-one relationship between T and S is detected when T includes a foreign key (different from its primary key) referencing S, and at most one tuple of T has the value of the primary key of each tuple of S. In this case, no explicit schema constraint assured the correctness of this cardinality assumption; however, considering that data warehouse systems are typically fed by data sources populated with a huge amount of data –hence, instances are representative of the application domain–, we can reasonably take it as true. A specific QVT transformation has been developed for detecting this pattern by integrating the algorithm proposed in [4]. Precisely, two queries are performed over T to count the number of non-null values of its foreign key with and without duplicates; the QVT transformation stores the results, compares them, and marks the foreign key as one-to-one if they are equal.

Foreign keys not marked as one-to-one are marked as many-to-one. In our running example, the foreign keys that link NationalSale and InternationalSale to Sale are marked as (strong) one-to-one, as well as the (weak) one that connects Organizer to Category. The other foreign keys are marked as many-to-one.

Calculating the in-degree of tables. A QVT transformation rule has been defined to calculate in-degree of tables. Note that a foreign key that has already been marked as one-to-one is not taken into account here, due to the possibility to navigate these relationships in both ways. Indeed, two tables marked as facts can be linked by a foreign key expressing a one-to-one relationship.

In our running example, table Order has in-degree 1, while Sale has in-degree 0 even if it has two incoming foreign keys (from NationalSale and InternationalSale, respectively), because these were marked as one-to-one.

Marking facts and measures. A table is marked as a fact if (1) its NIT and NAR are greater or equal to the thresholds, and (2) its in-degree is 0 or 1. The comparison is made by the QVT transformation presented in Fig. 5. Then, all numerical attributes of each table T marked as fact (excluding those belonging to the primary key of T) are marked as potential measures. In our example, Sale, Order, OrderDetail, and Product meet the first constraint, so they can be marked as facts. However, Product is not marked as a fact because its in-degree is 2 (i.e. the second constraint is not fulfilled).

Marking dimensions. For each table T marked as fact, its dimensions and the related hierarchies can be derived by following many-to-one relationships as normally done in current approaches (e.g., [5,1]).

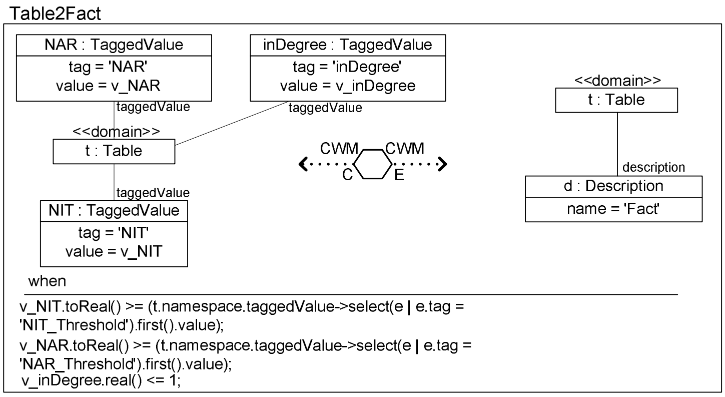


Fig. 5. QVT transformation for marking facts

Spawning analysis contexts. The aim of this phase is to create a set of models, each related to a possible analysis context, so as to generate every multidimensional solution implicitly contained in the relational data sources. This is done in two situations:

1. *Fact-dimension conflicts.* After the marking process, the marked rCWM model may present some configurations of marks that lead to inconsistencies in the multidimensional schema. These conflicts must be handled before creating the multidimensional representation of elements. Precisely, a marked rCWM model contains a conflict when a table is marked both as a fact and as a dimension. In our example, there is a conflict in the Order table. To overcome this problem, for each table T that has a conflict two rCWM models, corresponding to two different analysis contexts, are spawned: one where T is marked as dimension, one where it is marked as fact.
2. *Specialization.* When a table T marked as fact has a one-to-one foreign key referencing table S, we spawn two rCWM models: only S is marked as fact in the first one; S and T are marked as facts in the second one. For example, InternationalSale and NationalSale are both linked with one-to-one relationships to Sale. This leads to creating three rCWM models where: (1) only Sale is marked as fact, (2) Sale and InternationalSale are marked as facts, and (3) Sale and NationalSale are marked as facts.

In the end, the total number of rCWM models spawned depends on the number of conflicts and specializations in the original marked rCWM model. Precisely, the total number of rCWM models is $MN = (CN * 2) * \prod_{i=1}^{SN} SNT_i$ where CN is the number of fact-dimension conflicts, SN the number of specializations, and SNT_i the number of tables involved in the i-th specialization.

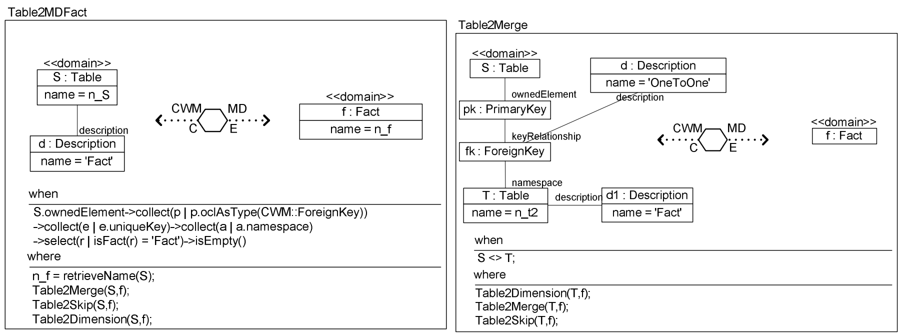
It is worth noting that an exponential number of rCWM models is obtained this way. In order to manage these high amount of models, our proposal can be easily integrated in the model-driven approach for data warehouse development proposed in [16,17], where the rCWM models can be reconciled with a conceptual schema previously defined from the information requirements of decision makers. A single multidimensional schema, that at the same time fits data sources and fulfills user requirements, is obtained this way. Due to space constraints, this reconciliation phase is not discussed in this paper.

3.3 Deriving Multidimensional Elements

The spawning phase creates one or more rCWM models. Two special patterns have been developed for handling special situations that can arise afterwards, namely (1) *skip* and (2) *merge*. Both share the same starting situation, i.e., two tables T and S marked as facts and such that T references S via a foreign key. The patterns are distinguished depending on the the mark applied to this foreign key.

1. When the foreign key is marked as many-to-one, a skip pattern is detected. In this case, T and its dimensions are not included in the multidimensional schema, so as to focus on the right granularity in each case. For example, the OrderDetail fact-marked table is skipped and Order is considered as fact. We recall that OrderDetail will be considered as fact in one or more other solutions.
2. A merge pattern is detected when the foreign key is marked as one-to-one. In this case, a fact is created whose dimensions and measures are the union of those belonging to T and S. For instance, Sale can be merged with NationalSale or InternationalSale to create facts for national and international analysis purposes, respectively.

These patterns are applied using QVT transformations, one of which (`Table2Merge`) is shown in Fig. 6b. In this merge transformation, an input pattern consisting of a table T marked as fact that refers S by means of a foreign key fk marked as one-to-one, leads to create a fact f (previously created from table S by means of the `Table2MDFact` transformation as shown in Fig. 6a). Importantly, according to the QVT transformations called in the WHERE clause,



```
when
S.ownedElement->collect(p | p.oclAsType(CWM::ForeignKey))
->collect(e | e.uniqueKey)->collect(a | a.namespace)
->select(r | isFact(r) = 'Fact')->isEmpty()
where
n_f = retrieveName(S);
Table2Merge(S,f);
Table2Skip(S,f);
Table2Dimension(S,f);
```

```
when
S <> T;
where
Table2Dimension(T,f);
Table2Merge(T,f);
Table2Skip(T,f);
```

(a) Obtaining facts (b) Merging facts

Fig. 6. QVT from a marked rCWM model to a multidimensional schema

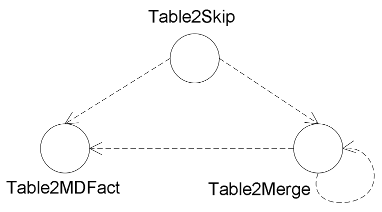


Fig. 7. Transformation execution order

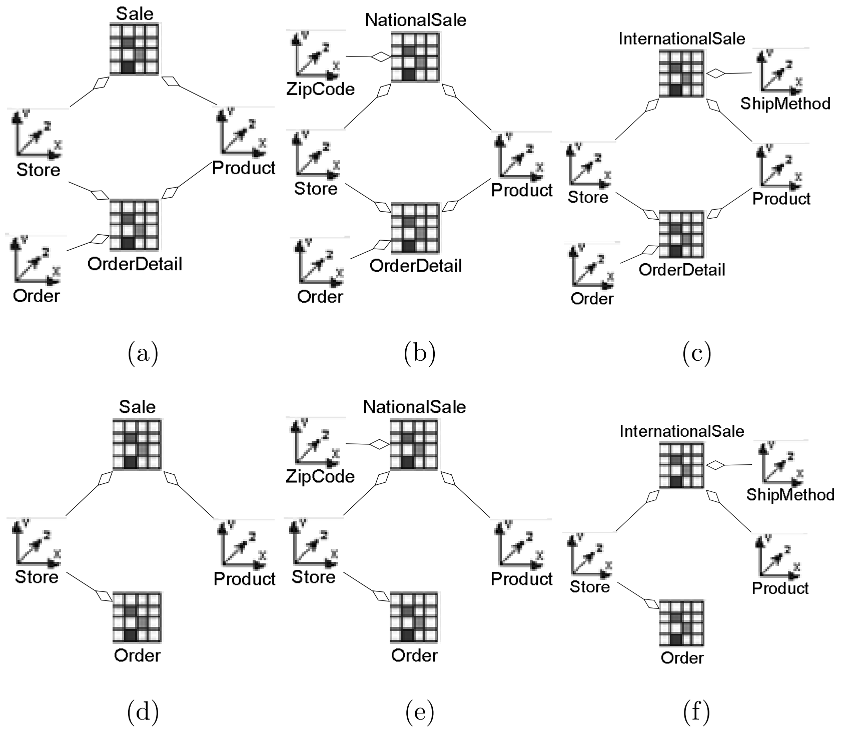


Fig. 8. Approach results over running example

the multidimensional counterparts of all the tables related to T will be related to f. Besides, when merge transformations are applied, the name of the table analyzed in the last merge transformation called is chosen as the fact name.

As to the order for applying transformations, the `Table2MDFact` transformation is executed first to create all facts, then special patterns are detected and applied by means of the QVT transformations called in the WHERE clause. The transformation flow is graphically represented in Fig. 7 using the approach defined in [18].

In Fig. 8 we present the solutions derived by applying our approach to the running example (measures and time dimensions are not shown for simplicity). The solutions in Fig. 8a, 8b, and 8c consider as facts OrderDetail and Sale in a general, national, and international analysis context respectively. The solutions in Fig. 8d, 8e, and 8f consider as facts Order rather than OrderDetail. As a whole, these solutions bring to light the full multidimensional potential of data sources; designers can then select the solution that best matches user requirements.

4 Conclusions and Future Work

Current approaches for data-driven conceptual design do not give designers a comprehensive and formal approach to detect facts. To fill this gap, in this paper we presented a model-driven approach for formalizing fact discovery in relational data sources by means of QVT transformations. Our approach is based on a set of heuristics relying on syntactical information derived from the data sources, thus guiding designers in the detection of multidimensional facts independently of their knowledge about the application domain. Remarkably, our approach has low computational complexity; the total processing time for the largest relational source schema we used for testing (about 130 tables and 140 foreign key constraints) is about 20 seconds.

The proposed model transformations have been implemented in the ECLIPSE[2] development platform. ECLIPSE is an open source project which has been conceived as a modular platform that can be extended by means of plugins in order to add more features and new functionalities. In that way, we have designed a set of modules encapsulated in a single plugin that provides ECLIPSE with capabilities for supporting our approach:

Relational module. It implements the relational metamodel contained in CWM.

Multidimensional module. The profiling mechanism of the *Unified Modeling Language* (UML) has been used to create multidimensional models.

Transformation module. It uses *mediniQVT*[3], a QVT transformation engine, in order to code and execute the mapping patterns.

Acknowledgments. This work has been supported by the QUASIMODO (PAC08-0157-0668) project from the Castilla-La Mancha Ministry of Education and Science (Spain). We would like to express our gratitude to personnel at *TAHE Fertilidad* (`http://www.tahefertilidad.es`) for their support during the development of this work.

References

1. Phipps, C., Davis, K.C.: Automating data warehouse conceptual schema design and evaluation. In: Proc. DMDW, pp. 23–32 (2002)
2. Jensen, M.R., Holmgren, T., Pedersen, T.B.: Discovering multidimensional structure in relational data. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, pp. 138–148. Springer, Heidelberg (2004)

[2] `http://www.eclipse.org`

[3] `http://projects.ikv.de/qvt`

3. Song, I.Y., Khare, R., Dai, B.: SAMSTAR: a semi-automated lexical method for generating star schemas from an entity-relationship diagram. In: Proc. DOLAP, pp. 9–16 (2007)
4. Alhajj, R.: Extracting the extended entity-relationship model from a legacy relational database. Inf. Syst. 28(6), 597–618 (2003)
5. Golfarelli, M., Maio, D., Rizzi, S.: The Dimensional Fact Model: A conceptual model for data warehouses. Int. J. Cooperative Inf. Syst. 7(2-3), 215–247 (1998)
6. Hüsemann, B., Lechtenbörger, J., Vossen, G.: Conceptual data warehouse modeling. In: Proc. DMDW, p. 6 (2000)
7. Böhnlein, M., von Ende, A.U.: Deriving initial data warehouse structures from the conceptual data models of the underlying operational information systems. In: Proc. DOLAP, pp. 15–21 (1999)
8. Moody, D.L., Kortink, M.A.R.: From enterprise models to dimensional models: a methodology for data warehouse and data mart design. In: Proc. DMDW, p. 5 (2000)
9. Romero, O., Abelló, A.: Multidimensional design by examples. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 85–94. Springer, Heidelberg (2006)
10. Mazón, J.N., Trujillo, J.: A model driven modernization approach for automatically deriving multidimensional models in data warehouses. In: Proc. ER, pp. 56–71 (2007)
11. Object Management Group: MOF 2.0 Query/View/Transformation, `http://www.omg.org/cgi-bin/doc?ptc/2005-11-01`
12. Soutou, C.: Relational database reverse engineering: Algorithms to extract cardinality constraints. Data Knowl. Eng. 28(2), 161–207 (1998)
13. Hopcroft, J.E., Tarjan, R.E.: Efficient algorithms for graph manipulation [h] (algorithm 447). ACM Commun. 16(6), 372–378 (1973)
14. Object Management Group: Common Warehouse Metamodel Specification 1.1, `http://www.omg.org/cgi-bin/doc?formal/03-03-02`
15. SAS Institute: Base SAS 9.1.3 Procedures Guide. Second edn. (2006)
16. Mazón, J.N., Trujillo, J., Lechtenbörger, J.: Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. Data Knowl. Eng. 63(3), 725–751 (2007)
17. Mazón, J.N., Trujillo, J.: A hybrid model driven development framework for the multidimensional modeling of data warehouses. SIGMOD Record 38(2), 12–17 (2009)
18. Meliá, S., Kraus, A., Koch, N.: MDA transformations applied to web application development. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, pp. 465–471. Springer, Heidelberg (2005)