Integration and Reuse of Heterogeneous Information: Hetero-Homogeneous Data Warehouse Modeling in the Common Warehouse Metamodel

Christoph Schütz  
Department of Business Informatics - Data & Knowledge Engineering, Johannes Kepler University, Linz, Upper Austria, Austria., schuetz@dke.uni-linz.ac.at  

Bernd Neumayr  
Department of Business Informatics - Data & Knowledge Engineering, Johannes Kepler University, Linz, Upper Austria, Austria., neumayr@dke.uni-linz.ac.at  

Michael Schrefl  
Department of Business Informatics - Data & Knowledge Engineering, Johannes Kepler University, Linz, Upper Austria, Austria., schrefl@dke.uni-linz.ac.at  

Follow this and additional works at: http://aisel.aisnet.org/amcis2012

Recommended Citation  
http://aisel.aisnet.org/amcis2012/proceedings/SystemsAnalysis/13

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2012 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Integration and Reuse of Heterogeneous Information: Hetero-Homogeneous Data Warehouse Modeling in the Common Warehouse Metamodel

Christoph Schütz  
Johannes Kepler University, Austria  
schuetz@dke.uni-linz.ac.at

Bernd Neumayr  
Johannes Kepler University, Austria  
neumayr@dke.uni-linz.ac.at

Michael Schrefl  
Johannes Kepler University, Austria  
schrefl@dke.uni-linz.ac.at

ABSTRACT

The corporate data warehouse integrates data from various operational data stores of a company. These operational data stores may be heterogeneous with respect to the represented information. The hetero-homogeneous data warehouse modeling approach overcomes issues associated with the integration of heterogeneous information from the operational data stores by featuring a generally homogeneous schema which may be interspersed with heterogeneities in well-defined portions of the data. In order to leverage the capabilities of existing business intelligence (BI) tools for the analysis of hetero-homogeneous information, the schema must comply with the metamodel of the particular BI tool. The Common Warehouse Metamodel (CWM) is a standard for data warehouse metadata which facilitates the reuse of data across multiple BI tools. In this paper, we present guidelines for modeling hetero-homogeneous data warehouses in the CWM. We demonstrate feasibility with a proof-of-concept prototype for the model-driven implementation of hetero-homogeneous data warehouses.

Keywords

Data warehouse, business intelligence, heterogeneous databases, metadata, conceptual modeling.

INTRODUCTION

The corporate data warehouse provides decision-makers with the information that is required for well-founded decisions, integrating data from various operational data stores of the different departments and local branches of a company. These data stores may be heterogeneous with respect to the represented information. For example, the production department has different requirements for their data than the accounting department. The data model of the corporate data warehouse reconciles the diverse data models of the operational data stores in a company.

In general, the integration of heterogeneous data models either results in the elimination of all heterogeneities or yields an overly complex integrated data model which preserves heterogeneities but is unfit for analysis purposes. Neither solution is particularly appealing. The elimination of heterogeneities leads to the loss of valuable information that could otherwise improve decision quality. Complex data models, on the other hand, preserve heterogeneities at the expense of clarity. These data models exceed the capabilities of the analyst who is overburdened with the sheer volume and complexity of the presented information. Existing modeling solutions do not adequately solve this integration issue.

The hetero-homogeneous approach as presented by Neumayr, Schrefl and Thalheim (2010) overcomes the limitations of other data warehouse modeling techniques with respect to the representation of heterogeneities. Hetero-homogeneous data warehouse models are homogeneous with respect to a globally agreed schema. This schema may be specialized for portions of the data provided the specialization does not violate the global schema. Depending on which portion of the data is analyzed, the analyst can either rely on the globally agreed, homogeneous schema or leverage an increased amount of information that is available specifically for the analyzed portion. This principle recursively applies to the portions of the portions of the data. For example, a data warehouse records monthly revenues of product sales by city. For U.S. sales, a refined schema is employed, recording revenues by store rather than city and, apart from revenues, recording the sold quantity. Whenever the analysis concerns only U.S. data, the analyst can rely on an increased amount of information.
In order to leverage the capabilities of existing business intelligence (BI) tools for the analysis of hetero-homogeneous information, the schema must comply with the specific metamodel that is employed by the particular BI tool. The Common Warehouse Metamodel (CWM) is an open industry standard for representing and managing warehouse metadata which facilitates the reuse of multidimensional data across multiple BI tools. The CWM as a standard for representing multidimensional schemas plays a major role in component reuse, which is a key issue in modern software engineering (Golfarelli, Rizzi and Turricchia, 2011). Real-world use cases confirm the suitability of the CWM to serve as a foundation for data warehouse metadata standardization (Melchert, Schwinn, Herrmann and Winter, 2005). The CWM easily maps to the proprietary metadata models of software vendors. In addition, many BI tools natively support the CWM either for exchange (IBM Infosphere Data Architect, SAS Data Integration, CA ERwin Data Modeler) or as their primary representation model (Pentaho Metadata). Consequently, many data warehouse modeling approaches base their concepts on the CWM (Prat, Akoka and Comyn-Wattiau, 2006) or rely directly on the CWM for vendor-neutral specifications of multidimensional data models (Paradillo and Mazón, 2012).

In this paper, we present guidelines for modeling hetero-homogeneous data warehouses in the CWM. We begin with a short introduction about data warehousing in general and the CWM in particular, followed by a review of the relevant literature. We then illustrate our modeling approach on a basic example. We demonstrate feasibility of our approach by providing a proof-of-concept prototype for the model-driven implementation of hetero-homogeneous data warehouses. This prototype builds on the existing management system for hetero-homogeneous data warehouses as introduced by Schütz (2010), extending this system with functionality for the export of CWM metadata in order to improve interoperability with other BI tools.

BACKGROUND

Data Warehousing, Online Analytical Processing, and the Common Warehouse Metamodel

Data warehouses organize strategic data for decision support using dedicated data models. Unlike operational databases, which support day-to-day business operations, data warehouses provide decision makers with information at an adequate level of detail (Elmasri and Navathe, 2007, p. 977). This information is commonly represented by multidimensional schemas. A multidimensional schema consists of (hyper-) cubes and dimensions. Dimensions consist of hierarchically ordered aggregation levels; cubes consist of cells. The cells of a cube represent business events of interest which are quantified by measures. For example, a cube may visualize product sales over time in various local markets (Figure 1). Each cell of such a sales cube might store the monthly revenues of a product model within a city or store.

Figure 1. A three-dimensional sales cube with heterogeneities in the USA region of the cube
Online Analytical Processing (OLAP) refers to the analyses that are performed on the data of a data warehouse. Among the most common OLAP operations are slice and dice, roll up and drill down. Slice and dice refer to the selection of cells from a base cube in order to obtain a sub-cube. This sub-cube may contain only the information that is needed for a particular analysis. Roll up and drill down refer to a change in granularity of the viewed data. Hierarchically organized dimensions allow for the analysis of data at various levels of granularity. For example, a cube may store the monthly revenues of products in French and U.S. cities (Figure 1). The French area manager may be interested only in the France portion of the cube, thus performing a dice operation on the cube to obtain only sales in France. Through summarization of revenues by year and product category, the analyst may roll up the data to obtain an aggregated view with a coarser level of granularity. Likewise, along with the dice operation to obtain a USA sub-cube, an analyst may drill down for an in-depth analysis to view the revenues by store rather than city.

In real-world applications, not all portions of a cube of data are uniform. Some portions of a cube may contain more information than others. Such heterogeneous cubes present a problem for modelers and analysts alike. Modelers are torn between a faithful representation of information and the definition of an understandable schema. Analysts cannot rely on the results of the analysis as availability of the necessary information cannot be guaranteed for any portion of the cube. For example, the sales cube in Figure 1 might contain additional data for the USA sub-cube. For U.S. sales, revenues are available per store rather than city. Furthermore, U.S. sales can be aggregated by states, a political entity that is non-existent in other countries. Apart from revenues, the USA sub-cube records an additional measure, namely the sold quantity (QtySold). The sold quantity, however, is recorded by product category, year, and city. The USA sub-cube is thus multi-granular, containing measures at multiple levels of granularity.

Throughout the remainder of this paper, we use the cube in Figure 1 as hypothetical example for illustration purposes. Real-world applications have shown, though, that similar kinds of heterogeneities actually occur in reality. Consider, for example, the data model of the German marketing research institute GfK (Bauer and Günzel, 2009), which features, among others, heterogeneous product hierarchies.

The analysis of data requires detailed knowledge about nature and structure of the analyzed data. Therefore, in order to effectively and efficiently analyze the data, BI tools require the schema of the analyzed data. This schema is metadata – data about data. Metadata, much like “ordinary” data, again follow a specific schema which is referred to as the metamodel.

Many BI tools employ their own proprietary metamodel. The Object Management Group (OMG), in an effort to harmonize the various proprietary models for warehouse metadata, promotes the Common Warehouse Metamodel (CWM) as an open industry standard. The CWM builds on the Unified Modeling Language (UML) and complies with the Meta Object Facility (MOF). UML and MOF are widely accepted standards for model-driven software development. The Extensible Markup Language (XML) and the XML Metadata Interchange (XMI) language serve as a universally understood exchange format for CWM metadata. A more detailed introduction to the CWM standard is given by Poole, Chang, Tolbert and Mellor (2003).

Related Work

Conceptual data warehouse models abstract from a concrete implementation and present a business-oriented view on the data. The Dimensional Fact Model (DFM) as proposed by Golfarelli, Maio and Rizzi (1998) is arguably the most popular conceptual modeling approach for data warehouses. The DFM emphasizes on facts and dimensions. Facts are business events of interest which are quantified by measures. The dimensions are hierarchically organized. Consequently, measures may be aggregated along the dimension hierarchies, thereby allowing for the analysis of facts on multiple levels of granularity. Most modeling approaches rely on similar modeling concepts. Pedersen, Jensen and Dyreson (2001) evaluate several data warehouse modeling approaches and identify key requirements for modern data warehouse modeling. Popular conceptual modeling approaches, however, fall short of these requirements.

Most notably, the accurate representation of heterogeneous information presents a tough challenge for modelers. Heterogeneities add considerably to the complexity of data warehouse models. Furthermore, heterogeneous models are prone to summarizability issues (Mazón, Lechtenbürger and Trujillo, 2009). Summarizability is an important property which ensures the correctness of aggregation operations (Lenz and Shoshani, 1997). Multidimensional normal forms for data warehouse dimensions have been proposed in order to avoid summarizability issues (Lehner, Albrecht and Wedekind, 1998; Lechtenbürger and Vossen, 2003). Similarly, integrity constraints allow to reason about summarizability within heterogeneous dimensions (Hurtado, Gutierrez and Mendelzon, 2005; Hurtado and Gutierrez, 2007). Apart from dimensions, heterogeneities may also occur in facts, yielding multi-granular data (Iflıkhar and Pedersen, 2010).

More recently, several conceptual data warehouse modeling approaches based on the UML or the Entity-Relationship (ER) model have been proposed which provide an increased support for complex modeling issues. Abelló, Samos and Saltor...
(2006) propose yet another multidimensional model (YAM²) based on the UML for modeling and querying data warehouses. Luján-Mora, Trujillo and Song (2006) develop a UML profile for conceptual data warehouse modeling. Malinowski and Zimányi (2006, 2008) introduce the MultiDimER model which builds on the concepts of the ER model. The MultiDimER model incorporates an inheritance mechanism for representing heterogeneous dimensions. Pinet and Schneider (2009) with their UML-based approach follow a similar principle for allowing heterogeneities in dimensions. Still, the UML/ER-based modeling approaches do not satisfactorily solve summarizability issues in heterogeneous dimensions and generally yield overly complex data warehouse models. Moreover, the UML-based approaches commonly focus on heterogeneous dimensions, thereby neglecting heterogeneous, multi-granular facts.

The hetero-homogeneous modeling approach of Neumayr et al. (2010) features a generally homogeneous data warehouse model which may be interspersed with heterogeneities in well-defined portions of the data. Sub-dimensions and sub-cubes may introduce additional information with respect to the entire dimension or cube, respectively, without violating the common global model. This feature proves advantageous especially when integrating data models of different data warehouses (Schütz, Schrefl, Neumayr and Sierninger, 2011). The hetero-homogeneous modeling approach allows for the preservation of heterogeneities during the process of model integration while reducing the complexity of heterogeneous data models.

Their support for model-driven implementation is a major advantage of UML-based conceptual modeling approaches. Prat et al. (2006) provide a full-fledged data warehouse design method based on the UML. This design method covers the transformation of the conceptual model into a logical model and a physical implementation. A similar design approach builds on the UML and the CWM (Mazón and Trujillo, 2008; Pardillo and Mazón, 2012). Kurze and Gluchowski (2010) give an overview of current approaches and hint future research directions for model-driven data warehouse development.

Rather than extending the UML, Neumayr et al. (2010) employ multilevel objects (m-objects) and multilevel relationships (m-relationships) for modeling hetero-homogeneous data warehouses. M-objects and m-relationships overcome the strict separation of schema and instance. This schema/instance duality allows for the introduction of a compact inheritance mechanism for dimension and fact schemas. Schütz (2010) presents a proof-of-concept prototype implementation for managing hetero-homogeneous data warehouses in a standard object-relational database. In this prototype implementation, the relational database automatically derives from the conceptual model with its m-objects and m-relationships. Despite providing basic export and analysis functionality, the system lacks a powerful exchange mechanism which preserves hetero-homogeneous information for other analysis tools.

**LOGICAL HETERO-HOMOGENEOUS MODELING IN THE COMMON WAREHOUSE METAMODEL**

In the CWM, the logical model describes the multidimensional structure of the data warehouse. The logical model consists of dimensions, hierarchies, and cubes as well as mappings between individual model elements. The logical model abstracts from a particular implementation, leaving the choice of the appropriate data structures to the physical model.

In this section, we present guidelines for modeling hetero-homogeneous dimensions and cubes in the CWM. We base our approach on Neumayr et al. (2010) and refer to Poole et al. (2003) as an authoritative guide for the development of CWM applications. Some details of CWM modeling, for example, certain types of mappings, which are not specific to hetero-homogeneous data warehouse modeling are mentioned only briefly or left out due to space considerations.

**Dimensions**

The dimensions set the context for the analysis. A dimension consists of a multitude of members which belong to the same real-world concept. These members are used to denote business events of interest. Attributes further describe the members of a dimension. In each dimension, a designated attribute identifies the members of the dimension. This attribute is referred to as the key of the dimension. For the key attribute, each member of the dimension must provide a unique value which no other member of that same dimension shares. The key attribute may be used to obtain a list of members of the dimension.

A dimension collects its members into aggregation levels. Some attributes are relevant for members at specific aggregation levels only. For that reason, attributes are attached to an aggregation level. Think of an aggregation level as a class. The class defines a set attributes; the instances assign values to these attributes. Similarly, the aggregation level defines a set of attributes; the members of the level assign values to these attributes. The members of a level, however, are not obliged to provide a value for each and every attribute that is defined for the level.

For example, dimension Location contains data about geographic entities (Figure 2). Geographic entities exist at various levels of aggregation, namely Country, Region, State, City, and Store. Each geographic entity has a unique name which is the key of the dimension. For geographic entities at level City, the dimension contains data about the mayor and the elevation.
The levels of a dimension are arranged in aggregation hierarchies. Within an aggregation hierarchy, the levels of a dimension are ordered from coarsest to most specific. The order of the levels determines the roll-up relationships between the levels. Each member of a level rolls up to a member of the precedent level. For example, given the primary hierarchy of dimension Location (Figure 3), level All is coarser than level Country and level Country is coarser than level City.

A dimension may have multiple aggregation hierarchies. Modelers commonly make use of this possibility in order to model alternative aggregation hierarchies for a dimension. For example, every city is located in a country and, in addition, every city is part of a region. Consequently, dimension Location – besides its primary hierarchy with levels City, Country, and Top (Figure 3) – has an alternative aggregation hierarchy with levels City, Region, and Top which is not shown in the examples.

Each hierarchy is homogeneous with respect to the levels, their relationships, and the attributes. While the dimension is inherently heterogeneous, defining for each level only optional attributes, the hierarchy imposes attributes and roll-up relation-
ships to its members. That is, each member of an aggregation hierarchy must provide a value for all attributes associated with its level and roll up to a member of the precedent level. Note that this behavior is not standard CWM. Rather, it is an important guideline for hetero-homogeneous modeling.

The possibility of defining multiple aggregation hierarchies for the same dimension also allows for the representation of hetero-homogeneous information. In case a branch of a dimension introduces heterogeneities to the model, the branch has its own aggregation hierarchy defined. The branch is homogeneous with respect to the information that is contained in its hierarchy; it may be heterogeneous with respect to other branches, though.

For example, the USA branch of dimension Location has additional aggregation levels. All U.S. cities roll up to a state, a political entity that does not exist in other countries. Also, the USA branch provides an additional level of detail underneath the City level: Level Store is the most specific aggregation level that is available for the whole USA branch. Furthermore, for all U.S. cities, data about the mayor is available in addition to the elevation. The USA branch thus introduces heterogeneities. Consequently, the USA branch has its own aggregation hierarchy defined (Figure 4).

This USA hierarchy concretizes the primary hierarchy of dimension Location. It does not contradict the information given in the primary hierarchy. After all, level City still has attribute Elevation and rolls up to level Country. The USA hierarchy, however, includes more information which could not be included in the primary hierarchy without rendering the primary hierarchy heterogeneous. The USA hierarchy, nevertheless, is homogeneous even after the inclusion of this additional information. The information may not be available for the whole dimension; it is for the entire USA branch, though.

**Cubes**

A cube is of arbitrary dimensionality and contains an arbitrary number of measures. Cubes are the most important modeling primitive as they organize the main data of interest – measures. For example, Figure 5 illustrates cube Sales of dimensions Product, Time, and Location. The cube has measures Revenue and QtySold.

In hetero-homogeneous CWM modeling, the cube itself does not assert any level of granularity for any measure nor does it guarantee that a measure is available for all portions of the cube. Rather, a cube defines what is to be expected at the most from the data of a cube.

In order to represent hetero-homogeneous cubes, the logical model borrows the concept of cube regions from the deployment model. In hetero-homogeneous modeling, cube regions represent portions of a cube with a homogeneous schema. Each cube region is single-granular, homogeneous with respect to the measures, and its schema applies only to a portion of the cube. Note that this is only a guideline for hetero-homogeneous modeling and not standard CWM behavior.

In case a sub-cube introduces heterogeneities with respect to the global schema, the sub-cube has its own cube region defined. The same principle as for hetero-homogeneous dimensions applies. Each cube region is homogeneous and guarantees
certain properties for the data of a sub-cube. The cube region, however, may be heterogeneous with respect to a global schema in that it concretizes this schema. The cube region may contain an arbitrary subset of the cube’s set of measures.

For example, cube Sales has three cube regions (Figure 6), each having a distinct set of measures and a single level of granularity. The primary cube region ProductTimeLocation defines the homogeneous schema for the entire cube. This cube region’s level of granularity is <Model, Month, City>, meaning that the measures of this cube region are recorded by product model, month, and city. The USA sub-cube records an additional measure and tracks revenues by store rather than city. For the USA sub-cube, two separate cube regions exist. The first cube region, ProductTimeUSA1, contains the measures that are recorded at the <Country, Year, City> level of granularity. The other cube region, ProductTimeUSA2, contains the measures that are recorded at the <Model, Month, Store> level of granularity. Notice that the definition that these cube regions are limited to the USA sub-cube cannot be done in the logical model but will rather be accomplished in the physical model.

PHYSICAL HETERO-HOMOGENEOUS MODELING IN THE COMMON WAREHOUSE METAMODEL

The physical model describes the representation of the data in a database. The physical model consists of tables as well as mappings between these tables and the logical model. While the logical model serves as a schema for the formulation of analytical queries, the data for answering these queries come from the tables that are defined in the physical model.

In this section, we propose a relational implementation for hetero-homogeneous data warehouses which considers the particularities of the logical model in the CWM. We rely on a variant of the widely-accepted star schema, consisting of dimension and fact tables. We further provide guidelines for mapping the logical hetero-homogeneous model to the physical representation of the data.

Dimension Tables

A dimension stores its data in a single dimension table. Each row of the dimension table represents a member of the dimension. The columns of the table represent the attributes of the dimension’s members. Two additional columns are introduced as auxiliaries for the implementation, namely Level and ID. These columns have no connection to the logical model and are not used for query formulation. Column Level holds the aggregation level of the dimension’s member that is represented by a row of the dimension table. Column ID serves as a surrogate key.

For example, the Location dimension table has columns All, Country, Region, State, City and Store for the key attribute of each level. The table further has columns Elevation and Mayor for the attributes of level City, and the table has columns Level and ID. The country France would be represented as a row with value ‘Country’ in column Level, given the fact that France is a country. The value in column ID must be an arbitrary yet unique value, possibly provided by a sequence. Furthermore, the France row would have value ‘Location’ in column All, ‘France’ in Country, and NULL values in State, City, Elevation, Mayor, and Store. The NULL values reflect the fact that some attributes are not applicable to all members of a dimension – the dimension is heterogeneous.

A surrogate key allows for the representation of dimension members at various aggregation levels within a single table. Without a surrogate key, the number of columns that identify a row would vary depending on the level of the represented dimension member. This surrogate key does not necessarily bear meaningful semantics; values for the surrogate key might well come from a sequence. Again, the surrogate key is not used for query formulation, but is only an internal auxiliary for the implementation.

The dimension tables are not normalized. In operational databases, the normalization of tables ensures non-redundant data sets. Large tables with many columns are split into smaller, redundancy-free tables in order to prevent the occurrence of update and delete anomalies. Anomalies, however, are not an issue in a data warehouse environment since updates and deletes of existing rows are rather uncommon. The smaller number of tables in a non-normalized schema, on the other hand, reduces the number of table joins and consequently improves query performance.

For example, given the Location dimension table, the French city of Paris would be represented as a row with value ‘City’ in column Level and again some unique value in column ID. Furthermore, the Paris row would have value ‘Paris’ in column City. The values in other columns would match the values in the France row. The table therefore contains redundancies. Splitting the dimension table into several smaller tables, for example, a separate table for level Country and level City, would prevent redundancies. This split, however, would also increase the number of joins required in analytical queries.

The analyst formulates queries over the logical model. The BI tool answers these queries using the physical model. Thus, the BI tool must be able to relate the logical model to the physical model. Explicit mappings formalize the relationships between
the dimensions in the logical model and the tables in the physical model. For example, a set of feature maps relates the attributes of level City of dimension Location to the corresponding columns of the dimension table (Figure 7).

Each hierarchy of a dimension presents a homogeneous view on the dimension table. The view of a hierarchy only contains columns for levels and attributes that are associated with that very same hierarchy. The view also lacks the Level column and does not have a surrogate key. Rather, the view of a hierarchy contains only members at a single aggregation level, namely the most specific level of the hierarchy. Likewise, the homogeneous view of a hierarchy does not contain any NULL values. The existence of NULL values would indicate the existence of heterogeneities within the hierarchy. The existence of heterogeneities in a supposedly homogeneous branch would violate the hetero-homogeneous approach. The heterogeneities should thus be removed from the hierarchy and their introduction be pushed further down the aggregation hierarchy.

For example, the view for the primary hierarchy of dimension Location includes only columns All, Country, and City for the levels of the hierarchy. The view also includes column Elevation for the only attribute that is shared by all members that are part of this hierarchy’s City level. Notice that including column Mayor in the view would yield NULL values since information about a city’s mayor is not available for every city; the view would not be homogeneous anymore. Furthermore, the view contains only the City-level rows of the dimension table. The following query characterizes the homogeneous view of the primary hierarchy of dimension Location:

```sql
01 SELECT 1.All, 1.Country, 1.City, 1.Elevation
02 FROM location l
03 WHERE 1.Level = ‘City’;
```

Similarly, the view of the USA hierarchy of dimension Location includes columns Country, State, City, and Store for the levels of the hierarchy. Besides column Elevation, the view also includes Mayor since information about the mayor is available for every city in the USA. The view contains only the Store-level rows of the dimension table. The following query characterizes the homogeneous view of the USA hierarchy of dimension Location:

```sql
02 FROM location l
03 WHERE 1.Level = ‘Store’ AND
04 1.Country = ’USA’;
```

The aggregation hierarchies map to the homogeneous views instead of the heterogeneous dimension tables. Two types of mappings are especially noteworthy. First, a `listOfValues` mapping allows to retrieving for each attribute the set of values that are actually assigned by the dimension’s members to the respective attribute. For the key attribute of a level, the `listOfValues` mapping retrieves the list of dimension members at this level. Second, an `immediateParent` mapping defines for each level the column that holds the names of the parent member of each member of the respective level.

Figure 8 illustrates mappings for the USA branch of dimension Location. A set of feature maps relates the attributes of the City level to the corresponding columns of the USA view. Notice that the feature maps are part of different classifier maps which in turn are linked to different structure maps. The feature map that relates attribute Parent of the hierarchy’s City level to column State of the USA view is part of the classifier map that is linked to the `immediateParent` mapping. The other mappings are `listOfValues`. 

Figure 7. Mapping dimension Location to its dimension table (level City only)
Fact Tables

A cube stores its data in a single fact table. This fact table has a column for each of the cube’s dimensions and for each of the cube’s measures. The dimension columns reference the table of the respective dimension. The measure columns record the values of the measures. Each row of the fact table represents a business event of interest at some level of granularity. The level of granularity is determined by the dimension columns. A fact table can be multi-granular and heterogeneous.

Figure 9 illustrates the star schema organization of cube Sales. The Sales fact table links to the surrogate key columns of the dimension tables of Product, Time, and Location. The first row in the Sales fact table records the revenues of the product model SonyBraviaKL46 in Paris in January 2012. It contains a NULL value in QtySold. The second row records the revenues of SonyBraviaKL46 in the SeattleStore1 in January 2012 and contains a NULL value in QtySold. Finally, the third row records the sold quantity of products in the Television category in the city of Seattle in the year 2012. Thus, the Sales fact table is heterogeneous and multi-granular.

Figure 8. Mapping the USA branch of dimension Location to its database view (level City only)
Each cube region presents a homogeneous, single-granular view on the heterogeneous, multi-granular fact table. Columns of the fact table that represent measures from other cube regions are disregarded. Furthermore, measures available at more specific granularities are rolled up to the required level of granularity, assuming that such a roll-up is possible without summarizability problems, which may not always be the case (see Lenz and Shoshani, 1997).

For example, the primary cube region of cube Sales contains only measure Revenue. Measure QtySold, which is available only for U.S. sales, is disregarded. The level of granularity is < Model, Month, City>, meaning that monthly revenues of product models are recorded by city. Since U.S. sales are tracked at a finer level of granularity, namely by store rather than city, the revenues must be aggregated in order for the view to remain single-granular. The following query characterizes the homogeneous, single-granular view of the primary cube region of cube Sales:

```
01 SELECT p.Model AS Product, t.Month AS Time, l.City AS Location,
02       SUM(s.Revenue) AS Revenue
03 FROM Sales s JOIN Product p ON s.Product = p.ID
04     JOIN Time t ON s.Time = t.ID
05     JOIN Location l ON s.Location = l.ID
07 GROUP BY p.Model, t.Month, l.City;
```

Measure QtySold is available only for U.S. sales and is recorded at the < Category, Year, City > level of granularity. The view for this cube region selects only the column for measure QtySold, disregarding measure Revenue, as it is tracked at a different level of granularity. The view further restricts the number of rows that are considered, selecting only U.S. sales. The following query characterizes the homogeneous, single-granular view of the USA cube region at the < Category, Year, City > granularity level:

```
01 SELECT p.Category AS Product, t.Year AS Time, l.City AS Location,
02       SUM(s.QtySold) AS QtySold
03 FROM Sales s JOIN Product p ON s.Product = p.ID
04     JOIN Time t ON s.Time = t.ID
05     JOIN Location l ON s.Location = l.ID
06 WHERE l.Country = 'USA'
07 GROUP BY p.Category, t.Year, l.City;
```

Measure Revenue is available at a finer granularity for U.S. sales only. While this heterogeneity was eliminated in the primary cube region, a separate view preserves this additional information in the USA cube region:

```
01 SELECT p.Model AS Product, t.Month AS Time, s.Store AS Location,
02       SUM(s.Revenue) AS Revenue
03 FROM Sales s JOIN Product p ON s.Product = p.ID
04     JOIN Time t ON s.Time = t.ID
05     JOIN Location l ON s.Location = l.ID
06 WHERE l.Country = 'USA'
07 GROUP BY p.Model, t.Month, l.Store;
```

Depending on which part of the cube is required for the analysis, a different view serves as the source for answering the query. The choice of what view will be used to answer a query depends on the context of the analysis. With the views being homogeneous, the analyst does not have to worry about heterogeneities when formulating the query.

The cube regions map to the homogeneous views instead of the heterogeneous, multi-granular fact table. Figure 10 illustrates mappings from the USA cube region at granularity level < Category, Year, City > to the corresponding view. The mappings are straight-forward; each attribute or measure from the cube region maps to a column of the same name.

![Figure 10. Mapping the USA cube region at granularity level < Category, Year, City > to its database view](image-url)
MODEL-DRIVEN IMPLEMENTATION OF HETERO-HOMOGENEOUS DATA WAREHOUSES

The presented guidelines for modeling logical and physical hetero-homogeneous data warehouse models can be automated. Without machine support, the definition of a logical and physical model in the CWM would be a tedious task for any modeler to complete. Rather, a modeler should rely on the conceptual modeling approach as presented by Neumayr et al. (2010) which offers a more intuitive approach towards hetero-homogeneous data warehouse modeling, using m-objects and m-relationships for the representation of dimensions and cubes. From this conceptual model, the CWM logical and physical models can be derived automatically.

Schütz (2010) presents a first approach towards the automatic derivation of a logical and physical data model from a conceptual hetero-homogeneous data warehouse model according to Neumayr et al. (2010). The management system of Schütz (2010) derives dimension and fact tables from the conceptual model. Furthermore, the system also holds the conceptual model itself for management and analysis purposes in object-relational tables. These tables may be accessed from the outside in order to gain insights about the conceptual model. These insights, in turn, may serve as the basis for the derivation of the logical and physical model in the CWM.

We provide export functionality for the management system of Schütz (2010) which consists of transformation routines in Java that extract from the database tables the hetero-homogeneous conceptual model and derive from this model the CWM logical and physical models. For the representation of CWM metadata in Java, our export functionality relies on the Pentaho Metadata libraries. These libraries implement the CWM specification in Java. Furthermore, the Pentaho Metadata libraries allow for the representation of CWM metadata using the XMI language. XMI data is well understood by various BI tools and therefore well-suited for the exchange of models.

SUMMARY & FUTURE WORK

In this paper, we presented an approach for modeling hetero-homogeneous data warehouses in the CWM. Figure 11 illustrates the proposed modeling approach. The conceptual model abstracts from the actual implementation and provides a high-level perspective on the modeling domain. M-objects and m-relationships are well-suited for the conceptual representation of hetero-homogeneous dimensions and cubes (Neumayr et al. 2010). In order to use this conceptual model with various BI tools, the conceptual model must be translated into a more widely understood standard data model. This standard data model is the CWM, with the XMI language as its primary representation format.

The logical hetero-homogeneous CWM model consists of dimensions and their hierarchies as well as cubes and their cube regions. Hierarchies provide homogeneous schemas for particular branches of a heterogeneous dimension. Likewise, cube regions provide homogeneous schemas for particular portions of a multi-granular, heterogeneous cube.

Figure 11. Hetero-homogeneous data warehouse modeling in the CWM in a nutshell

1 The current version of the prototype software is available on http://hh-dw.dke.uni-linz.ac.at/
homogeneous CWM model consists of multiple tables. Dimension and fact tables contain the actual data. The heterogeneous dimension tables in the physical CWM model map to corresponding dimensions of the logical model. The hierarchies of these dimensions, in turn, map to homogeneous views on the heterogeneous dimension tables. Likewise, the cube regions map to homogeneous views on the heterogeneous fact tables. Notice that the cubes in the logical model do not map directly to the fact tables. The CWM provides cube regions for mapping the cubes of the logical model to the tables of the physical model; cube regions are therefore not truly part of the logical model. In order to represent hetero-homogeneous cubes, however, cube regions are employed in the logical model.

Our approach to hetero-homogeneous data warehouse modeling in the CWM facilitates the integration of heterogeneous data sources and allows for the reuse of hetero-homogeneous data models in various BI tools. We stress that our approach to hetero-homogeneous modeling is complementary to the modeling approach of Neumayr et al. (2010) and the corresponding prototype presented by Schütz (2010). Rather than providing an alternative modeling approach and implementation, we extend the existing prototype system with a powerful export mechanism. Existing infrastructure may thus be reused without losing the benefits of hetero-homogeneous data models. Still, future work will have to address the following issues:

- Examine the usability of hetero-homogeneous CWM models in various BI tools.
- Define a transformation process compliant with the OMG’s model-driven architecture from the hetero-homogeneous conceptual model to the logical and physical CWM models.
- Provide a more concise graphical notation for hetero-homogeneous models in the CWM.

ACKNOWLEDGMENTS

This work was partly funded by a Marshall Plan Scholarship granted by the Austrian Marshall Plan Foundation to Christoph Schütz while away from Johannes Kepler University for a research stay at Portland State University.

REFERENCES


