Dieses Dokument ist eine Zweitveröffentlichung (Postversion) /

This is a self-archiving document (accepted version):

Bernhard Jaecksch, Wolfgang Lehner

The Planning OLAP Model: A Multidimensional Model with Planning Support

Erstveröffentlichung in / First published in:

Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse, 29.08. - 02.09.2011. Springer, S. 14–25. ISBN 978-3-642-23544-3.

DOI: http://dx.doi.org/10.1007/978-3-642-23544-3_2

Diese Version ist verfügbar / This version is available on:

https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-830666

SLUB Wir führen Wissen.

TECHNISCHE UNIVERSITÄT DRESDEN

QUCOSA Quality Content of Saxony

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

The Planning OLAP Model - A Multidimensional Model with Planning Support

Bernhard Jaecksch and Wolfgang Lehner

TU Dresden, Institute for System Architecture,
Database Technology Group, 01062 Dresden, Germany
bernhard.jaecksch@mailbox.tu-dresden.de,
wolfgang.lehner@tu-dresden.de

Abstract. A wealth of multidimensional OLAP models has been suggested in the past, tackling various problems of modeling multidimensional data. However, all of these models focus on navigational and query operators for grouping, selection and aggregation. We argue that planning functionality is, next to reporting and analysis, an important part of OLAP in many businesses and as such should be represented as part of a multidimensional model. Navigational operators are not enough for planning, instead new factual data is created or existing data is changed. To our knowledge we are the first to suggest a multidimensional model with support for planning. Because the main data entities of a typical multidimensional model are used both by planning and reporting, we concentrate on the extension of an existing model, where we add a set of novel operators that support an extensive set of typical planning functions.

1 Introduction

With the rise of decision-support-systems and the use of data-warehouses in many modern companies, the research community devised various models to support multidimensional analysis in the process of On-Line Analytical Processing (OLAP) [4]. The common data entities to model such multidimensional data are so called cubes consisting of a set of orthogonal dimensions and mostly numerical fact-data characterized by the values of the different dimensions. The main aspect of OLAP is the navigation through and aggregation of multidimensional data. The models provide an algebra of operators that often contains typical operators of relational algebra transferred to the multidimensional scenario and extended by navigational operators to group, select and aggregate data, also termed as slice/dice and roll-up/drill-down operations.

Business planning is an important task in many companies where business targets are defined for future periods to provide specific guidelines for current operations and a comparison whether goals have been reached or not. As such, planning is an important part of many practically used decision-support-systems. However, to our knowledge, none of the existing multidimensional models supports planning functionality. We strive to overcome the limitation of existing

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

models to support planning functionality as part of OLAP. As the basic data entities are the same for planning and reporting, we build on an existing OLAP model an extend its set of operations with novel operators to support a list of typical planning functions.

The paper is structured as follows: in the next section we describe related work in the field of OLAP models as well as the multidimensional model that serves as foundation for our OLAP model with planning support. Section 3 introduces a list of typical planning functions by example. Our novel operators to support planning are introduced in Section 4 where we show how to express the planning functions with the set of extended operators. We finish with a conclusion in Section 5 providing an outlook for an implementation of our model.

2 Foundation and Related Work

Starting with the Data Cube operation by Gray et al. [6] as an extension of SQL, a wealth of multidimensional models have been proposed. Similar to the Data Cube, the first models by Li et al. [9] and Gysses et al. [7] were extensions to the relational model. The field of statistical databases also dealt with the quantitative analysis of large amounts of scientific data and, faced with similar problems, suggested different models. Prominent candidates are the Summary Tables model by Ozsoyoglu et al. [10] and the graphical model for Statistical Object Representation (STORM) by Rafanelli et al. [12]. While all these models, divide the data into *qualifying* and *quantifying* information, most modern models are base on the concept that the qualifying information defines a multidimensional space represented by a cube where each axis is called a dimension. The quantifying information at the intersection points, called measures or facts, is characterized by the dimensions. Typical and often cited representatives are the Multidimensional Database Model by Agrawal et al. [1], the F-Table Calculus by Cabibbo et al. [3], the Cube Operations model by Vassiliadis et al. [13], the Multidimensional Object model by Lehner [8] and the Cube Data model by Datta et al. [5]. Vassiliadis provides a good classification and survey of these models in [14]. The suitability of the models to implement a practical and complex data-warehouse scenario was evaluated by Pedersen et al. [11] according to an extensive set of typical requirements such as explicit hierarchies, multiple and flexible hierarchies per dimension, symmetric treatment of dimensions and measures and explicit aggregation semantics. Since none of the previous models fulfilled all requirements they suggested their own model, the Extended Multidimensional Data model (EMDM). As this model satisfies all of the above requirements we considered it a suitable foundation for our planning extensions.

The basic EMDM model entity is a multidimensional object $MO = (S, F, Dim, R)$, which is a four-tuple consisting of an *n-dimensional fact schema* S, a set of facts F, a set of *dimensions* Dim and a set of corresponding *fact-dimension relations* R that map the facts to elements of the dimensions. A key aspect of the model is that everything that characterizes a fact is regarded dimensional. That includes measures and as such dimensions and measures are treated

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

symmetrically. An n-dimensional fact schema S is a two-tuple (FS, D) with FS describing a fact type and D being a set of dimension types $D = \{T_i, i = 1..n\}$. Each dimension type T itself is a four-tuple $(C, \prec_T, \top_T, \bot_T)$, where C is a set of category types $\{C_j, j = 1..k\}$ of T that form a partial ordering $\prec_T$ with $\top_T$ and $\bot_T$ as the top and bottom elements of the ordering. There is always a single top element that contains all other elements. For certain category types it often makes sense to aggregate them. To support the different aggregation types in the model, three different classes of aggregation functions exist: constant c, average functions ϕ and sum functions Σ. For these classes an ordering exists such that $c \subset \phi \subset \Sigma$. For each dimension type T the model provides a function that determines the aggregation type for a category type. A dimension Dim_i has a dimension type T that is defined in the fact schema of an MO as explained in the previous section. $Dim_i = (Ca, \prec)$ is a two-tuple with Ca being a set of categories $\{Ca_j\}$ and $\prec$ a partial ordering on all dimension values e in each category Ca_j with $Type(e) = C_j$. Furthermore, all values e in the dimension Dim_i are smaller than value $\top$ and the most granular values are contained in category $\bot_T$. To establish a connection between facts and dimensions, fact-dimension relations are introduced. A fact-dimension relation R is a set of two-tuples $\{(f, e)\}$ where f is a fact and e is a dimension value. Therefore, the fact is characterized by the dimension value e. Values from different dimension categories can determine the same fact. Also it must be ensured in the model that each fact in R is characterized by at least one dimension value. Thus, if there is no suitable dimension value to characterize a fact, the value $\top$ is used. Based on these entities an algebra with a list of operators is part of the model. As basic operators, all operations from relational algebra like *selection*, *projection*, *rename*, *union*, *difference* and *join* are adopted to operate on MOs. In addition, the *aggregate formation* operator allows to built aggregates and group facts. Typical OLAP operators like roll-up, *drill-down*, *SQL-like aggregation* and *star-joins* are expressed in terms of the basic operators.

3 Common Planning Functions by Example

The example used throughout the paper has the schema shown in Figure 1, consisting of 5 dimensions with dimension types *Article, Location, SellDate, Price* and *Quantity*. They characterize the *Sale* of a product. Usually, *Price* and *Quantity* would be considered as measures, so we call them the measure dimensions. Each dimension consists of different categories that form one or more hierarchies per dimension. For the two dimensions *Article* and *Location* Figures 2 and 3 show dimension values and their partial ordering.

The measure dimensions contain numerical values, where quantities are integral numbers and prices are values taken from real numbers. In Table 1 we list the facts for our example in column 1 and each entry represents a fact. The other columns represent the dimensions that characterize the fact and the value(s) in each row represent the dimension value(s) forming, together with the fact, (an) element(s) of the fact-dimension relations.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

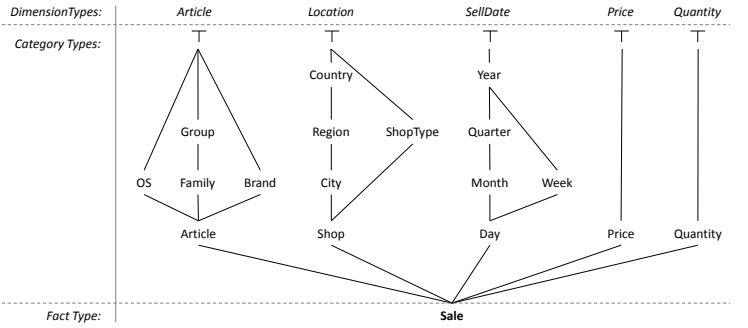

Fig. 1. An example schema

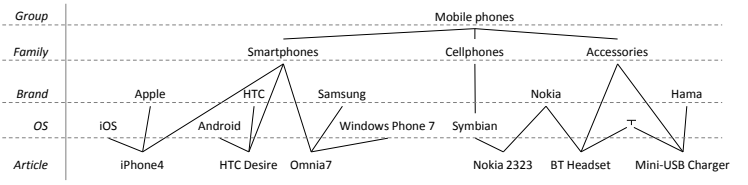

Fig. 2. Dimension values and partial ordering for dimension *Article*

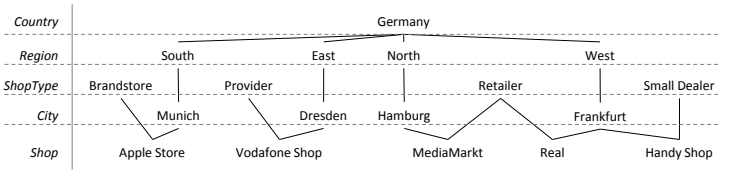

Fig. 3. Dimension values and partial ordering for dimension *Shop*

The following example introduces a list of common planning functions in a typical business planning scenario. Our model should be able to express each of these functions. Assume a company that sells phones and accessories to shops and retailers. Our example schema shows a multidimensional object with a list of facts that capture sales for year 2010. It is the task of a controller to plan sales quantities and prices for year 2011.

Step 1. As a first step he wants to base his plan on the values of the previous year and therefore he needs a planning function that **copies** data from 2010 into year 2011. The new MO would now contain twice as much facts as before.
Step 2. Because of recent market trends the company decides to sell only smartphones in 2011 and therefore a **delete** planning function deletes all standard cellphone sales from 2011.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

Table 1. Lists of facts in the example schema

Fact	$R_{Article}$	$R_{Location}$	$R_{SellDate}$	R_{Price}	$R_{Quantity}$
f_1	iPhone4,iOS, Apple,Smartphones	AppleStore,Munich, BrandStore	2010-05-06, 05, Q2, 2010	699.00	50
f_2	Desire,Android, HTC,Smartphones	Vodafone Shop, Dresden,Provider	2010-04-23, 04, Q2, 2010	479.00	35
f_3	Omnia 7,Windows Phone 7,Samsung, Smartphones	Media Markt, Hamburg,Retailer	2010-12-14, 12, Q4, 2010	389.00	10
f_4	2323,Symbian, Nokia, Cellphones	Real,Frankfurt, Retailer	2010-01-11, 01, Q1, 2010	79.95	110
f_5	BT Headset,Nokia, Accessories	HandyShop,Dresden, Smalldealer	2010-03-27, 03, Q1, 2010	24.55	70
f_6	USB Charger,Hama, Accessories	Real,Frankfurt, Retailer	2010-08-13, 08, Q3, 2010	12.99	45

Step 3. For the year 2011 the prices are lowered by 5% compared to the previous year. Therefore the planner calls a **revalue** planning function to apply these changes. Furthermore, he wants to know the planned revenue that is based on sales quantities and price.
Step 4. For a each retailer the planner requests the estimated quantity of sold items from the sales person that is responsible for this customer. These quantities are now entered to the plan at aggregated customer level and must be distributed to individual facts using a **disaggregation** planning function.
Step 5. Finally, the controller wants to use the current data of 2010 and the planned data of 2011 to predict a sales quantity trend for 2012. He requires a planning function that calculates a **forecast** and generates a set of forecasted facts.

After all these steps, the complete plan data is created and can be used for reports and comparisons with the actual data of 2011. The list of planning functions that was involved includes copy, delete, calculate expressions to revalue quantitative values, disaggregation of new values from an aggregated level to the most granular fact level and forecasting new values.

4 An OLAP Model for Planning

In the following section we extend the EMDM and develop our novel model algebra to support planning. One major aspect of planning is that it changes fact data. All the models existing so far make the assumption that fact data is a read only set of values and all operators have navigational, i.e. read only semantic. With planning new facts will be created or existing facts are manipulated. Therefore, the novel operators for planning must support this. From the list of basic planning functions shown in Section 3 not all require a separate operator. Similar to the original model, where typical OLAP operators like roll-up and

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

drill down are expressed in terms of the basic aggregation formation operator, we only need a few basic operators to keep the extended algebra simple and minimal.

4.1 Basic Planning Operators

Value Mapping. An important basic operation, e.g. for the copy function, is *value mapping*. When new plan data has to be generated one can copy facts from a previous year. As a result, for a set of facts, one or more dimension values change. For example, to copy along the time dimension the values must change from one year to another. The mapping operator takes as input a set of mapping functions, which map a combination of source dimension values to corresponding target values. As fact-dimension relations are the glue between facts and dimensions, the mapping operator modifies these relations. We formally denote the mapping operator as:

Definition 1. *Value mapping* $\gamma\left[\{m_r, r = 1..s\}\right](MO) = (S', F', Dim', R')$ *takes a set of mapping functions* m_r*, which have the form* $m_r(e_1, .., e_i, .., e_n) \mapsto e_i'$*, with*

- $S' = S$, $F' = F$, $Dim' = Dim$
- $R' = \{R_i', i = 1..n\}$
- $R_i' = \{(f', e_i') | f' \in F' \wedge e_i' = m_r(e_1, .., e_i, .., e_n) \wedge e_1 \rightarrow_1 f, .., e_i \rightarrow_i f, .., e_n \rightarrow_n f \wedge e_i' \rightarrow_i f'\}$

Intuitively a mapping function m_r is applied to a fact-dimension relation R_i and maps the input value to itself or, for matching values, to a different dimension value. It is important that the mapping function is aware of dimension hierarchies and provides a complete mapping from one part of the hierarchy lattice structure to another. If we consider the *SellDate* hierarchy of our example schema and want to map from the year 2010 to the year 2011, a mapping function $f_{2010 \mapsto 2011}$ must provide a mapping for all dimension values that are in the partial ordering below 2010 to the respective values in the hierarchy below 2011. Thus *Q1-2010* would be mapped to *Q1-2011* and so on.

Duplication. The value mapping operator from the previous section modifies fact-dimension relations, but the set of facts is not changed. To introduce new facts, as it is necessary for a copy function, we add the duplication operator to the model, which duplicates the facts of an MO. The resulting MO' has identical dimension structure and fact-dimension relations with the exception that for each fact in MO there is a new fact in MO' that is characterized by the same dimension attributes and values.

Definition 2. *Duplication* $\tau(MO) = (S', F', Dim', R')$ *with*

- $S' = S$
- $F' = \{f' | \exists f \in F \wedge e_1 \rightarrow_1 f, .., e_n \rightarrow_n f \wedge f' \neq f \wedge e_1' \rightarrow_1 f_1', .., e_n' \rightarrow_n f_n' \wedge e_1 = e_1' \wedge \ldots \wedge e_n = e_n'\}$
- $R' = \{R_i', i = 1..n\}$
- $R_i' = \{(f', e_i') | \forall (f, e_i) \in R_i \wedge f' \in F' \wedge e_i' \in Dim_i'\}$

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

Disaggregation. A typical planning function is to enter an aggregated value for a group of facts and then calculate how it distributes to the individual fact values that contribute to the aggregated value. The disaggregation can be viewed as the reverse operation to the aggregation. In contrast to the drill-down operation, the disaggregation operation defines a new sum value and *changes* all contributing values accordingly. We define the disaggregation operator similar to the aggregation formation operator as the inverse operator α^{-1}. As input it takes a set of dimension values, that define the aggregation level where the new sum value is entered. Additional parameters are a distribution function g^{-1} and an aggregate function g that determines how the values are aggregated. Finally, a new dimension value e_{new} is given as well as the index t of the target dimension and the index r of a reference dimension. It is allowed that $t = r$, in which case the new value is distributed according to the original fractions dimension Dim_t.

Definition 3. *Disaggregation is* $\alpha^{-1}\left[e_1, .., e_n, g^{-1}, g, e_{new}, t, r\right](MO) = (S', F', Dim', R')$, *where*

- $S' = S$, $F' = F$, $Dim' = \{Dim_i, i = 1..n \wedge i \neq t\} \cup \{Dim'_t\}$
- $Dim'_t = (Ca'_t, \prec'_t)$, $\prec'_t = \prec_{t_{|Dim'}}$
- $Ca'_t = \{Ca'_{tj} \in Dim_t | Type(Ca'_{tj}) = \top_{Dim_t} \vee (Type(Ca'_{tj}) = \bot_{Dim_t} \wedge e'_{tj} = g^{-1}(e_{rj}, e_{new}, e_{old}) \wedge e_{rj} \in Ca_{rj} \wedge Type(Ca_{rj}) = \bot_{Dim_r} \wedge SUM(Group(e'_1, .., e'_n)) = e_{new} \wedge e_{old} = g(Group(e_1, .., e_n)) \wedge (e'_1, .., e'_n) \in Ca'_1 \times \cdots \times Ca'_n \wedge (e_1, .., e_n) \in Ca_1 \times \cdots \times Ca_n)\}$
- $R' = \{R'_i, i = 1..n \wedge i \neq t\} \cup \{R'_t\}$, $R'_i = \{(f', e') | f' \in F' \wedge e' \in Dim'_i\}$
- $R'_t = \{(f', e'_i) | \exists (e_1, .., e_n) \in Ca_1 \times \cdots \times Ca_n \wedge f' \in F' \wedge e'_i \in Dim'_t \wedge e'_i = g^{-1}(e_{ri}, e_{new}, e_{old}) \wedge \forall e_i \in Dim_t \exists e_{ri} \in Dim_r\}$

The fact-schema S' of MO' is the same as that of the original MO since only values are changed and no dimensions are added or removed. The set of facts is the same, too, because the disaggregation operator does not introduce new facts. It only maps the facts for the target dimension to new values. The set of dimensions is again taken from the original MO, but the target dimension Dim'_t changes in the sense that intuitively the new dimension values are calculated based on the new sum and the fractions of the given reference dimension Dim_r. The category attributes Ca'_{tj} of the target dimension are either the top attribute, or they are in the class of the most granular attribute and their new dimension values are calculated using the distribution function g^{-1}. The distribution function calculates the new dimension values e'_{tj} using the new sum e_{new} as input, the old reference aggregate value e_{old} and the respective dimension value e_{rj} of the reference dimension. The aggregate value e_{old} is obtained by applying the reference aggregate function g to the grouping of $Group(e_1, .., e_n)$ at the level of the given input dimension values. Finally, the fact-dimension mapping R'_t is adapted such that the facts are now mapped to the new dimension values calculated by the distribution function. This includes, the requirement that for each fact-dimension mapping in the target dimension there exists a fact-dimension mapping in the reference dimension.

By allowing arbitrary functions for the distribution function g^{-1} and the reference aggregate function g, different types of distribution can be achieved. For

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

example, a typical distribution function that calculates the new fraction based on the percentile of the reference value from the old sum value is $g^{-1}(e_r, e_{new}, e_{old}) = e_r * e_{new}/e_{old}$ together with $g = SUM$. The reference dimension can be the same as the target dimension. For a uniform distribution, the reference aggregate function should be $g = COUNT$ and for a constant distribution $g^{-1}(e_r, e_{new}, e_{old}) = e_{new}$. The following example illustrates how disaggregation works: we distribute a new article quantity of 384 to all sales facts in *Germany* for the year *2010*. The input MO contains all 5 dimensions *Article, Location, SellDate, Price* and *Quantity* and is not restricted. The parameters for the disaggregation are:

$$\alpha^{-1}\left[\top_{Article}, Germany, 2010, \top_{Price}, \top_{Quantity}, g^{-1}, g, e_{new} = 384, t = 5, r = 5\right](MO) = MO'$$

The distribution function g^{-1} is the standard function explained in the previous section and the reference aggregate function g is SUM. The target dimension *Quantity* now contains the new dimension values that would result in the new sum 384 when the reverse operation, i.e. the aggregate formation, would be applied to MO'. The disaggregation affects all facts $f_1, .., f_6$ in the example and the fact-dimension mapping $R_{Quantity}$ would change from

$$\{(f_1, 50), (f_2, 35), (f_3, 10), (f_4, 110), (f_5, 70), (f_6, 45)\}$$

to

$$R'_{Quantity} = \{(f_1, 60), (f_2, 42), (f_3, 12), (f_4, 132), (f_5, 84), (f_6, 54)\}$$

Calculated Dimensions. Another cornerstone of planning is to calculate various expressions on multidimensional data. We therefore allow expressions on multidimensional objects. Since everything is a dimension in the model and the facts are objects that are described by dimension values, such an expression is an operation on dimension values. We realize this within our model by defining a *calculated dimension* $\overline{Dim}$ of type $T = (C_j, \prec_T, \top_T, \bot_T)$ similar to basic dimensions as a two-tuple $\overline{Dim} = (Ca, \prec)$. The set Ca contains the categories of the dimension and $\prec$ is the partial ordering on all dimension values. The difference is, that a category attribute $Ca_i \in \overline{Dim}_{n+1}$ is now defined in terms of an expression where the operands are category attributes of other dimensions $Ca_i = \otimes(Ca_j, Ca_k)$ with $Ca_j \in Dim_r$, $Ca_k \in Dim_s$ and $\otimes$ being an arbitrary binary operator from the following list $\{+, -, *, /, \wedge, \vee\}$. In the same manner expressions can contain arbitrary scalar functions and operators by extending the definition of a calculated category type to the general form $Ca_i = \otimes(\{Ca_j\})$ where $Ca_j \in Dim_r, j = 1..m, r = 1..n+1, i \neq j$ and $\otimes$ is an arbitrary unary, binary or n-nary operator or scalar function applied to a number of category attributes. It is possible that the expression references other category attributes of the calculated dimension. This is useful for example to add constant to the expression by defining a category attribute that only has one dimension value and reference it in other expressions. To calculate such an expression we add a calculated dimension to a multidimensional object using the following operator:

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

Definition 4. *The add dimension operator* $+\left[\overline{Dim}_{n+1}\right](MO) = (S', F', Dim', R')$ *takes as input a multidimensional object MO and a calculated dimension* $\overline{Dim}_{n+1}$ *where*

- $S' = (FS', D')$, $D' = \{T'_i, i = 1..n\} \cup \{T_{n+1}\}$, $T'_i = T_i$, $F' = F$
- $\underline{Dim}' = \{Dim'_i, i = 1..n\} \cup \{Dim_{n+1}\}$, $Dim'_i = Dim_i$
- $\overline{Dim}_{n+1} = (Ca_{n+1}, \prec)$
- $Ca_{n+1} = \{Ca_{n+1,i} = \otimes(\{Ca_{rj} | Ca_{rj} \in Dim_r, j = 1..m, r = 1..n\})\}$
- $R' = \{R'_i, i = 1..n\} \cup \{R'_{n+1}\}$, $R'_i = R_i$
- $R'_{n+1} = \{(f', e'_{n+1,i}) | f' \in F' \wedge e'_r i \rightarrow f' \wedge e'_{n+1,i} = \otimes(e'_{ri}) \wedge e'_{ri} \in Ca_r\}$

To calculate an expression between values of two different MOs, first a join operator should be applied to create a combined MO and then a calculated dimension containing the expression is added.

As an example we will calculate the *revenue* for our mobile phone data. Therefore we add a dimension $Dim_{revenue} = (Ca, \prec)$ with $Ca = (Revenue, \top)$, $Revenue = Qty * Price$ and add this dimension to our multidimensional object $+[Dim_{Revenue}](MO)$. The resulting MO' now has an additional Revenue dimension where the dimension values of the *Revenue* category attribute are calculated for each fact as the product of the *Quantity* and *Price* dimension values.

Forecast. The need for a *forecast* operator is directly motivated by the respective forecast planning function. Besides copying values or enter plan values manually, it is often useful to use a forecasting function fc to project trends of historical data into the future. For the forecast operator this means creating a set of new fact values for a category attribute Ca_t (most often from a time dimension). An ordering $O[Ca_t]$ is required for all dimension values e_{ti} of type Ca_t. Let $O[Ca_t] = 1..m$ with $O[Ca_t](e_{ti}) < O[Ca_t](e_{tj}), i \neq j, i = 1..m, j = 1..m$ if e_{ti} is smaller than e_{tj} in terms of ordering O. The forecast function $f(F, O[Ca_t], tgt, k)$ can be an arbitrary forecasting algorithm like exponential smoothing or ARMA models [2]. It takes as input a set of facts F, an ordering $O[Ca_t]$ for these facts based on a given dimension Dim_i with $Ca_t \in Dim_t$, an integral number tgt which specifies the number of new facts it should produce and an index $k = 1..tgt$ that specifies which of the forecasted values it should return. We can now write the following definition for the forecast operator:

Definition 5. $\phi[f, O, Ca_t, Ca_v, tgt](MO) = MO' = (S', F', Dim', R')$ *with the fact schema staying the same* $S' = S$*, the set of facts is extended by tgt new facts and*

- $F' = F \cup \{f''_j | j = 1..tgt \wedge \top_1 \rightarrow_1 f''_j \wedge \ldots \wedge \top_i \rightarrow_i f''_j \wedge \ldots \wedge \top_n \rightarrow_n f''_j \wedge e'_t \rightarrow_t f''_j \wedge e'_v \rightarrow_v f''_j \wedge i \neq t \wedge i \neq v\}$
- $Dim' = Dim$, $R' = \{R'_i | i = 1..n \wedge i \neq t \wedge i \neq v\} \cup \{R'_t, R'_v\}$
- $R'_i = R_i \cup \{(f''_j, \top_i) | j = 1..tgt \wedge f''_j \in F'\}$
- $R'_t = R_t \cup \{(f''_j, e'_t) | j = 1..tgt \wedge f''_j \in F' \wedge O[Ca'_t](e'_t) = max(O[Ca'_t](e_k)) + j \wedge k = 1..n\}$
- $R'_v = R_v \cup \{(f''_j, e'_v) | j = 1..tgt \wedge f''_j \in F' \wedge e'_v = f(F, O[Ca_t], tgt, j)\}$

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

In essence the forecast operator produces l new facts, which are mapped to an ordered set of dimension values such that the new facts are mapped to the l successors of the last dimension value from the existing facts. Furthermore, for a given (measure) dimension each new fact is mapped to its projected new value according to the prediction of the forecasting algorithm.

4.2 Expressing Typical Planning Functions

The following section lists typical planning functions that are necessary to support planning applications. For each of these functions we describe it in terms of operators of our novel Planing-OLAP model.

Delete. The delete operator deletes fact values from a multidimensional object. We make no distinction here between an MO' where facts have only been filtered and actual deletion. Therefore, the planning operator delete can be expressed with the selection operator. Let $MO = (S, F, Dim, R)$ and p an arbitrary predicate, that selects the values for deletion, then $\sigma\left[\neg p\right](MO) = MO' = (S', F', Dim', R')$ results in MO' that only contains the *not*-deleted values.

Copy. When we introduced the *value mapping* operator, we already emphasized that copying is an important part of planning to set a starting point for subsequent planning operations with data based on historic values. The copy operator can be expressed in terms of the value mapping basic planning operation combined with the duplication. The new MO' is created by applying a mapping to an MO^{Copy} that contains duplicates of the original facts. Let $M = \{m_r, r = 1..s\}$ be a set of mapping functions, then copying is defined as: $\cup(\gamma\left[M\right](\tau(MO)), MO) = MO' = (S', F', Dim', R')$.

Disaggregation. The disaggregation planning function can be directly mapped to the disaggregation operator of the model.

Revalue. The revalue planning function is used to change a set of values according to a formula. To execute such calculations within our Planning-OLAP model, we use calculated dimensions.

Forecasting. Similar to the disaggregation planning function, the *forecasting* planning functions has a direct representation as an operator in our Planning-OLAP model and can therefore expressed with a call to this operator.

5 Impact and Conclusion

When comparing the Planning-OLAP model with traditional OLAP models then the distinction is, that the latter is based purely on read operations whereas the planning operators write or generate data. As planning has a simulation character, it is often the case that generated data is continuously adjusted until a

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

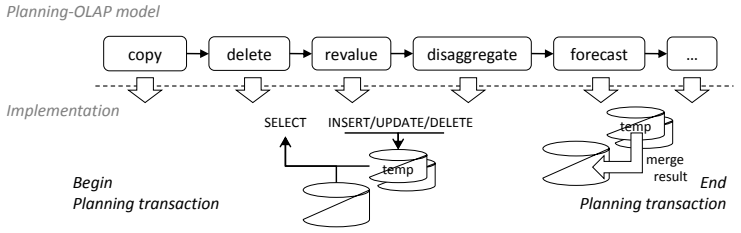

Fig. 4. Implementation scheme for the Planning-OLAP model

final result is obtained. As such it can be viewed as a long running transaction containing a mixture of read (*roll-up, drill-down, slice and dice*) and write (*disaggregate, copy, revalue, forecast and delete*) operations. While the transaction bracket is not necessary for the read-only traditional OLAP, it makes sense for the Planning-OLAP model where only the final result of a planning transaction should become visible and persistent. The scheme in Figure 4 outlines how this usage paradigm of the Planning-OLAP model can be mapped to a relational system using SQL. At the begin of a planning transaction one or more temporary tables are created that will contain the modifications of the current transaction and final results are merged into the original tables at the end of the transaction. Each operator is mapped to a combination of SELECT and INSERT/UPDATE/DELETE statements that modify the temporary tables. For example the disaggregation operator has to update every measure value that contributes to the overall sum. However, as the disaggregation operator contains possibly complex distribution logic, the orchestration of these statements to yield the correct result must either be done by the application or may be encapsulated in a stored procedure. Clearly, a direct integration of this functionality into the database system as a native operator would facilitate standardized and application independent behavior and allow for optimizations. This is similar for other operations such as value-mapping and forecasting. Therefore, we argue that in the future, these operations should be first-class citizens in a database system, equal to many navigational OLAP operators that are already supported natively by major database systems today.

Although, there exists a wealth of OLAP models in the literature, none of the existing models incorporated planning functionality. Many requirements have been formulated for OLAP modeling to support real-world scenarios. While many requirements are already met by some of the models, none of it explicitly supported planning, which is a vital part of OLAP today. We proposed a novel Planning-OLAP model that uses the Extended Multidimensional Data Model (EMDM) as a foundation. By introducing a set of novel planning operators our model is capable of supporting an extensive list of standard planning functions, which we illustrated by examples. Since the Planning-OLAP model contains operators that change data according to complex semantics, the challenge on the implementation level is to have planning operators as first-class citizens within a database system.

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden

Final edited form was published in "Data Warehousing and Knowledge Discovery: 13th International Conference. Toulouse 2011", S. 14–25, ISBN 978-3-642-23544-3
http://dx.doi.org/10.1007/978-3-642-23544-3_2

References

1. Agrawal, R., Gupta, A., Sarawagi, S.: Modeling Multidimensional Databases. In: Proc. of 13th. Int. Conf. on Data Engineering ICDE, vol. 7, p. 11 (1997), http://eprints.kfupm.edu.sa/51421/
2. Box, G., Jenkins, G.: Time Series Analysis: Forecasting and Control (1970)
3. Cabibbo, L., Torlone, R.: Querying Multidimensional Databases. In: Database Programming Languages, pp. 319–335. Springer, Heidelberg (1998), http://www.springerlink.com/index/f76731r05m5u662j.pdf
4. Codd, E., Codd, S., Salley, C.: Providing OLAP (On-Line Analytical Processing) to User-Analysis: An IT Mandate (1993), http://www.citeulike.org/user/MoritzStefaner/article/4937436
5. Datta, A.: The Cube Data Model: a Conceptual Model and Algebra for On-line Analytical Processing in Data Warehouses. Decision Support Systems 27(3), 289–301 (1999), http://linkinghub.elsevier.com/retrieve/pii/S0167923699000524
6. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In: ICDE, pp. 152–159 (1996)
7. Gyssens, M., Lakshmanan, L.V.S.: A Foundation for Multi-Dimensional Databases. In: Proceedings of the International Conference on Very Large Data Bases, pp. 106–115. Citeseer (1997), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.46.7255&rep=rep1&type=pdf
8. Lehner, W.: Modeling Large Scale OLAP Scenarios. In: Advances in Database TechnologyâĂŤEDBT 1998, p. 153 (1998), http://www.springerlink.com/index/1VR766FUCVW7NY4T.pdf
9. Li, C., Wang, X.S.: A Data Model for Supporting On-Line Analytical Processing. In: Proceedings of the Fifth International Conference on Information and Knowledge Management - CIKM 1996, vol. 199, pp. 81–88 (1996), http://portal.acm.org/citation.cfm?doid=238355.238444
10. Ozsoyoglu, G., Ozsoyoglu, Z., Mata, F.: A Language and a Physical Organization Technique for Summary Tables. In: Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data, pp. 3–16. ACM, New York (1985), http://portal.acm.org/citation.cfm?id=318899
11. Pedersen, T., Jensen, C., Dyreson, C.: A Foundation for Capturing and Querying Complex Multidimensional Data. Information Systems 26(5), 383–423 (2001), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.6209&rep=rep1&type=pdf
12. Rafanelli, M.: A Functional Model for Macro-Databases. ACM SIGMOD Record 20(1), 3–8 (1991), http://portal.acm.org/citation.cfm?id=122050.122051&coll=GUIDE&dl=ACM&idx=J689&part=periodical&WantType=periodical&title=ACMSIGMODRecord
13. Vassiliadis, P.: Modeling Multidimensional Databases, Cubes and Cube Operations. In: Proceedings of Tenth International Conference on Scientific and Statistical Database Management (Cat. No.98TB100243), pp. 53–62 (1998), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=688111
14. Vassiliadis, P., Sellis, T.: A Survey on Logical Models for OLAP Databases. SIGMOD Record 28, 64–69 (1999)

Provided by Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden