Modeling multidimensional database

Author: Dhananjay Patil
Organization: Evaltech, Inc.
Evaltech Research Group,
Data Warehousing Practice.
Date: 07/02/04
Email: erg@evaltech.com

Abstract:
The Purpose of Multidimensional modeling is to provide the business analyst with a tool to get quick answers on totals and averages from an attribute in a dataset. Moreover, the user could switch quickly to totals on sub-group level.
This white paper includes description logics, language, concept language and semantic of cube operators, description of an object-centered framework for multidimensional data models to overcome common vocabulary and common understanding of multidimensional problems.

Intellectual Property / Copyright Material
All text and graphics found in this article are the property of the Evaltech, Inc. and cannot be used or duplicated without the express written permission of the corporation through the Office of Evaltech, Inc.
Summary

Most multidimensional model makes a distinction between measures and dimensions. Measures are variables for which totals and averages are to be calculated and dimensions are variables that indicating groupings. The business analyst should precisely indicate which "measures" and "dimensions" are to be included in his multidimensional model. Such variables are then set in a multidimensional model. This is highly specialized work, which is often undertaken by an IT department.

In order to consolidate and analyze huge amounts of data, aggregation and roll-up play an important role. Data is often summarized at various levels and on various combinations of attributes. Therefore, queries are more complex and may take long time to complete. To face up to this problem, a great deal of effort has been invested in developing techniques for optimizing queries involving views and aggregate functions.

Unfortunately, a unifying framework for multidimensional databases is still missing. Hence different multidimensional models, operators, techniques, etc. cannot be compared to each other or evaluated. Furthermore, there is a certain lack of common vocabulary and common understanding of multidimensional data models.

The purpose of this white paper, to propose an object-centered framework for multidimensional data models to overcome this drawback.

It is a first step towards the development of a framework which

- Is equipped with well-defined semantics and
- Allows the precise definition of relevant reasoning services and problems.

This is a prerequisite for the investigation of these problems with respect to their complexity and the comparison of different reasoning techniques.

This article include following topics.

- Introduction to modeling multidimensional database
- Description logics
- The language and
- Basic definition
- The concept language
- Semantics of cube operators
- Conclusion
Introduction

Multidimensional modeling has its focus on providing totals and averages on group level and sub-group level. Switching between totals on sub-group level and group level can be done very quickly. Sophisticated tools are in use that generates such outcomes within a very short time-span. Hence, the performance of such tools is far better than the performance of any SQL statement that can be used.

The main strength of multidimensional databases is their ability to view, analyze, and consolidate huge amounts of data.

In general, multidimensional databases provide two categories of tools:

- Tools for integration, efficient storage and retrieval of large volumes of data
- Tools for viewing and analyzing data from different perspectives

These tools allow interactive querying of data and their analysis, often referred to by "navigation" since this way of querying seems much more intuitive than classical ad-hoc queries.

Instead of presenting tables to the user, data is presented in the shape of so called data cubes which using operators to cut out pieces from large can manipulate cubes change the granularity of dimensions, turn cubes, etc. Because of these functionalities, multidimensional databases play an important role in Decision Support Systems, for On Line Analytical Processing, and in Data Warehousing.

Following tables shows the basic terminology and its description in dimension modeling.

<table>
<thead>
<tr>
<th>Key word</th>
<th>Description/Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measures</td>
<td>Variable for which total and average to be calculated</td>
</tr>
<tr>
<td>Dimension</td>
<td>Variable that indicating grouping</td>
</tr>
<tr>
<td>Data cube</td>
<td>Data is presented in the shape</td>
</tr>
<tr>
<td>Cube</td>
<td>Manipulation by using operators to cut out into to no of pieces of large data cubes.</td>
</tr>
<tr>
<td>Multidimensional modeling</td>
<td>Cube</td>
</tr>
<tr>
<td>Cell</td>
<td>In a data cube, each axis is associated with a dimension (e.g., time, space, or products) and its according values. Then, points in the cube called cell.</td>
</tr>
</tbody>
</table>
Presentation of object-centered Framework

Description of logics and family of formalism

Useful for unifying framework for object-centered representation formalisms

These formalisms are equipped with well-defined semantics and sound, and complete reasoning algorithms.

The main characteristic of these formalisms is that problems like satisfiability, containment or consistency are effectively decidable.

These formalisms equipped with a kind of interface to specific “concrete domains” (e.g., integers, strings, reals), and built-in predicates.
### Example (A) for object-centered framework description

Consider the following table of a database that contains a report of heavy-duty battery sales. It presents the number of heavy-duty battery sold with respect to a heavy-duty battery’s model, (construction-) year, and color.

<table>
<thead>
<tr>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>2002</td>
<td>White</td>
<td>80</td>
</tr>
<tr>
<td>B1</td>
<td>2002</td>
<td>Black</td>
<td>45</td>
</tr>
<tr>
<td>B1</td>
<td>2003</td>
<td>White</td>
<td>60</td>
</tr>
<tr>
<td>B1</td>
<td>2003</td>
<td>Black</td>
<td>72</td>
</tr>
<tr>
<td>B2</td>
<td>2002</td>
<td>White</td>
<td>65</td>
</tr>
<tr>
<td>B2</td>
<td>2002</td>
<td>Black</td>
<td>90</td>
</tr>
<tr>
<td>B2</td>
<td>2003</td>
<td>White</td>
<td>70</td>
</tr>
<tr>
<td>B2</td>
<td>2003</td>
<td>Black</td>
<td>95</td>
</tr>
<tr>
<td>B3</td>
<td>2002</td>
<td>White</td>
<td>45</td>
</tr>
<tr>
<td>B3</td>
<td>2002</td>
<td>Black</td>
<td>32</td>
</tr>
</tbody>
</table>

Trying to compare the sales of white heavy-duty battery in 2003, it becomes obvious that it is difficult to extract and analyze information from this table. There is an alternative way of representing the same data, which overcomes this problem of jumping between rows. That representation is commonly known as a data cube. It is a cube with three intersecting 2D cross tabs. In this representation, each axis is associated with a dimension that is a column of a relational table. Elements within a dimension are called positions. Points in a data cube are called cells, and each cell is associated with the corresponding element of the column unit. Then, the cube is said to have dimensions model, (construction) year, and color, and to have the measure unit. The data in this representation is more organized; hence it is more easily accessible than the organization offered by a relational table. Note that the cube representation is only possible because the number of units sold is uniquely determined by a heavy-duty battery’s model, its construction year and its color. Multidimensional databases are designed for ease and performance in manipulating and analyzing huge amounts of complex data; hence values of dimensions or measures can be aggregated, decomposed, or combined to new values. This corresponds to what is commonly known as data consolidation.

The set of operators for data consolidations is claimed to be minimal and powerful enough to be used for all interesting queries on data cubes. It contains the following operators:

- Push allows to convert dimensions into measures,
- Pull allows to create a new dimension from a specified measure,
- Destroy allows to remove a dimension that has a single value in its domain,
- Restrict removes from the cube those values of a given dimension that do not satisfy a stated condition.
- Join allows to relate information in two cubes,
- Merge allows merging cubes by aggregating the values of some dimensions into new values of the same dimension and by aggregating the according measures.
What are Description logics?

Description logics (also called concept or terminological logics) are a family of formalisms designed to represent the taxonomic and conceptual knowledge of a particular application domain on an abstract, logical level.

They are equipped with well-defined, model-theoretic semantics and most of them have strong expressible power. Furthermore, the interesting reasoning problems such as containment and satisfiability are, for most description logics, effectively decidable. Description logics are built around concepts, which are interpreted as classes of objects in the domain of interest, and roles, which are interpreted as binary relations on these objects. Description logic is mainly characterized by a set of constructors, which allow to build complex concepts and roles from atomic ones. For example, from atomic concepts Human and Female and the atomic role child we can build the concept (Human and forall child.Female), which denotes the set of all Human, whose children are all instances of Female. Here, the constructor **and** denotes conjunction between concepts, while **forall** denotes (universal) value restriction. A knowledge base in a description logic system is made up of two components:

- the Tbox is a general schema concerning the classes of individuals to be represented, their general properties and mutual relationships;
- the Abox contains a partial description of a particular situation, possibly using the concepts defined in the Tbox. It contains descriptions of (some) individuals of the situation, their properties and their interrelationships. Retrieving information in a knowledge representation system based on description logics is a deductive process involving the schema (Tbox) and possibly an instantiation (Abox). In fact, the Tbox is not just a set of constraints on possible Aboxes, but contains intentional information about classes. This information is taken into account when answering queries to the knowledge base. The following reasoning services are the most important ones provided by knowledge representation systems based on description logics.

<table>
<thead>
<tr>
<th>Representation of Description logics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Concept satisfiability:</strong> Given a knowledge base and a concept C, does there exists at least one model of the knowledge base assigning a nonempty extension to C?</td>
</tr>
<tr>
<td><strong>Subsumption:</strong> Given a knowledge base and two concepts C and D, is D more general than C? That is, is in all models of the knowledge base each instance of C also an instance of D?</td>
</tr>
<tr>
<td><strong>Knowledge base satisfiability:</strong> Are an Abox and a Tbox consistent with each other? That is, does the knowledge base admit a model?</td>
</tr>
<tr>
<td><strong>Instance checking:</strong> Given a knowledge base, an object o, its possibly incomplete description, and a concept C, is o an instance of C in all models of the knowledge base?</td>
</tr>
</tbody>
</table>
The language

The syntax and semantics of a description logic for the representation of multidimensional data is explained as follows on the base of the one presented in (Baader & Hanschke 1991) and extends it by a constructor that captures functional dependencies (Borgida & Weddell 97) and a set of operators for the handling of data cubes.

- **Basic Definition**
  
  Multisets = \{1,1\}
  Simple Set = \{1\}

  In a multiset an individual can occur more than once i.e. aggregation.

- **Definition 1 (multisets)**
  
  A multiset (or bag) on a set \(S\) is a subset \(M\) of \(S \times N\) such that for every \(a\) element of \(S\) there is a non-negative integer \(n\) (the number of occurrences of \(a\) in \(M\)) such that \((a, m)\) is in \(M\) if and only if \(m < n\). The set of all multisets on \(S\) is denoted by \(M(S)\).

- **Definition 2 (concrete domain)**
  
  A concrete domain \(D = (\text{dom}(D), \text{pred}(D), \text{funct}(D), \text{agg}(D))\) consists of:
  
  - the domain \(\text{dom}(D)\),
    
    A set of predicate symbols \(\text{pred}(D)\), where each predicate symbol \(P \in \text{pred}(D)\) is associated with an arity \(n\) and a \(n\)-ary relation \(P^D \subseteq \text{dom}(D)^n\),
  
  - A set of 2-ary function symbols \(\text{funct}(D)\), where each function symbol \(f \in \text{funct}(D)\) is associated with a function \(f^D : \text{dom}(D)^2 \rightarrow \text{dom}(D)\), and
  
  - a set of aggregation symbols \(\text{agg}(D)\), where each aggregation symbol \(\Sigma \in \text{agg}(D)\) is associated with an aggregation function \(\Sigma^D : M(\text{dom}(D)) \rightarrow \text{dom}(D)\).

\[
\text{ALC}(D) = \text{Extension of description logic}
\]

In ALC, concepts can be built using Boolean operators (i.e., and, or, not), and value restrictions on those individuals associated to an individual via a certain role (binary relation). These include existential restrictions like in (exists has_child.Girl) as well as universal restrictions like (forall has_child.Human). Additionally, in ALC (D), (abstract) individuals, which are described using ALC, can now be related to values in a concrete domain, like, for example, the integers or strings. This allows us to describe, for example, persons whose savings balance are higher than their yearly income, by Human and (savings > y_income).

**Conclusion:** It is shown that all inference problems for ALC (D) are decidable provided that \(D\) does not contain any aggregation function and \(n\)-ary functions, and satisfies some additional very weak conditions.
The Concept Language

ALC (D) extended with a constructor that captures functional dependencies and operators on cubes specified at the extensional level. The language resulting from the extension of ALC (D) with a functional dependency operator and cube operators at the extensional level will be called ALC (D)\textsuperscript{fd}.

- Complex concepts are constructed using certain operators.
- These concepts can be used to specify the terminology in a so-called TBox.
- This TBox can be viewed as an encyclopedia in which the meaning of certain concepts is defined using other concepts (which are possibly defined themselves in this TBox).
- Then, a specific situation can be described in a so-called ABox, possibly using the concepts defined in the TBox.
- In the ABox, some individuals describe their properties and their interrelationships.

Definition 3 (Syntax of ALC(D)\textsuperscript{fd} - concepts)

Let $N_C$, $N_R$, and $N_F$ be disjoint sets of concept, role, and feature names, and let $D$ be an admissible concrete domain. Then each concept name is a concept, and complex concepts are defined inductively by the following rules, where $C$, $D$ are concepts, $R$ denotes a role name or feature name, $P \in \text{pred}(D)$ is a n-ary predicate name, $u_1, ..., u_n$ are feature chains (A feature chain $u = f_1 ... f_n$ is a composition of features), $f, f_1, ..., f_n$ are feature names. Then the following expressions are also concept terms.

- $C \text{ or } D$ (disjunction), $C \text{ and } D$ (conjunction), and $\text{not}(C)$ (negation),
- $\text{exists } R.C$ (exists-in restriction) and $\text{forall } R.C$ (value restriction),
- $P \left( u_1, ..., u_n \right)$ (predicate restriction),
- $[\text{fd } C \{ f_1, ..., f_n \}]$ (Functional dependency).

A terminology (or TBox) $T$ is a (finite) set of concept definitions, each of the form $A = C$ where $A$ is a concept name and $C$ a complex concept. Here restriction to those terminologies where each concept name $A$ occurs at most once on the left hand side of a concept definition, and which does not contain definitional cycles.
An Abox $A$ is a (finite) set of assertions. Given a set of individual names $N_i$, assertions are of the following forms,

$$a:C, (a, b):R, (a, b):f, (a, x):f, (x_1, ..., x_n):P,$$

$$a = \text{rename}(b, f, g), a = \text{destroy}(b, f), a = \text{restrict}(b, C), a = \text{join}(b, c),$$

$$a = \text{Join}((b, c, <n_1, o_1, f_1, g_1>, ..., <n_m, o_m, f_m, g_m>), a = \text{aggr}(b, f, \Sigma, g).$$

for (possibly complex) concepts $C$, feature names $f, f_i, g, g_i, n_i$, role name $R$, function names $o_i \in \text{funct}(D)$, an aggregation function $\Sigma \in \text{agg}(D)$, (abstract)

individuals $a, b, c \in N_i$, a $n$-ary predicate name $P$, and elements of concrete domain $x, x_1, ..., x_n$.

The semantics of these constructs is now defined in a model-theoretic way.

**Definition 4 (Semantics)**

Let $D$ be an admissible concrete domain. The semantics is then given by an interpretation $I = (\Delta^I, ..., I)$, which consists of an

(abstract) interpretation domain $\Delta^I$ disjoint from the concrete domain $\text{dom}(D)$, and an interpretation function $I$. The evaluation function $I$ associates each concept $C$ with a subset $C^I \subseteq \Delta^I$, each role $R$ with a binary relation $R^I \subseteq \Delta^I \times \Delta^I$, and each feature name $f$ with a partial function $f^I$ from $\Delta^I$ into $(\Delta^I \cup \text{dom}(D))$.

To satisfy the following equations:

$$(C \text{ and } D)^I = C^I \cap D^I$$

$$(C \text{ or } D)^I = C^I \cup D^I$$

$$(\text{not } C)^I = \Delta^I - C^I$$

$$(\forall R. C)^I = \{x \in \Delta^I \text{ such that for all } y, \text{ if } (x, y) \in R^I \text{ then } y \in C^I\}$$

$$(\exists R. C)^I = \{x \in \Delta^I \text{ such that there exists } y, \text{ with } (x, y) \in R^I \text{ and } y \in C^I\}$$

$$P(u_1, ..., u_n)^I = \{x \in \Delta^I \text{ such that } ((u_1)^I(x), ..., (u_n)^I(x)) \in P^D\}$$

$$[\text{fd } C \{f_1 \ldots f_n\} f] = \{x \in \Delta^I \text{ such that for all } y \in C^I, \text{ if } (f_1^I(x) = f_1^I(y) \land \ldots \land f_n^I(x) = f_n^I(y)) \}$$

then $f^I(x) = f^I(y)$
A concept C is satisfiable iff there exists an interpretation I such that $C^I \neq \{\}$. Such an interpretation is called a model of C. A concept C is subsumed by a concept D (written $C \subseteq D$) iff $C^I \subseteq D^I$ holds for each interpretation I.

An interpretation I satisfy a TBox T iff I satisfy each concept definition in T, and it satisfies a concept definition $A=C$ iff $A^I=C^I$.

If $(a, b) \in R^I$ or $f^I(a)=b$, we say that b is an R-filler (resp. f-filler) of a.

Interpretations of ABoxes, additionally, associate each individual name a to an element of $\Delta^I$, in such a way that $a^I \neq b^I$ holds for two different individual names a, b. Again I satisfies an Abox A iff I satisfies each assertion in A, that is to say.

$$a^I \in C^I \quad \text{for each } a:C \in A$$

$$(a^I, b^I) \in R^I \quad \text{for each } (a, b):R \in A$$

$$f^I(a^I) = b^I \quad \text{for each } (a, b):f \in A$$

$$f^I(a^I) = x^I \quad \text{for each } (a, x):f \in A$$

$$(x^I_1, ..., x^I_n) \in P^D \text{ for each } (x_1, ..., x_n):P \in A$$

The semantics of assertions containing operators like destroy or join is more difficult and will first be illustrated by examples and then given formally. An ABox A is said to be consistent with a TBox T iff there exists a model for both, that is an interpretation I satisfying A and T.

Note that the description of individuals in an ABox needs not to be complete: A model of an ABox might have more elements than those explicitly mentioned, and the individuals mentioned in the ABox might have more properties than those explicitly stated in the ABox. This is due to the so-called Open World Assumption.

**Example (A) (cont.)** Using ALC (D)$^f$ cubes containing information on heavy-duty battery sales can be described by the following concept definition:

$$\text{batterysales} = \text{forall has\_cell.}(\exists \text{model}.\text{Model and exists} \text{year}.\text{INTEGER and exists color}.\text{STRING and exists unit}.\text{INTEGER and fd batterysales } \{\text{model, year, color, unit}\}$$

where has\_cell is a role name and model, year, color, and unit are feature names. Model can be seen as a primitive concept containing the different models of batteries. Concrete domains needed in this example are INTEGER and STRING.
Semantics of Cubes Operators

The operators for which to define the semantics make sense only for "cubes", which have been introduced only informally so far. In general, a cube is an object, which is associated to cells which are all of similar form.

The following concept cube describes this fact of being of the same form.
It is defined in such a way that, if a cell of an instance of cube has an f-filler for some feature name f, then all other cells have also f-fillers.

cube = and, \( \in NF(\{ \exists \text{has}_f \text{cell}.(\exists f.\text{ALL} \text{or Top}_C(f)) \Rightarrow (\forall \text{has}_f \text{cell}.\exists f.\text{ALL}) \}) \)

where C \( \Rightarrow D \) is used as shorthand for \((\text{not}(C) \text{ or } D)\) and ALL is a shorthand for the "universal" concept \((A \text{ or not}(A))\) (\(A\) is a concept name). Given that-for each specific application-the set of feature names (NF) is finite, cube is a concept and describes cubes according to our intuition. For the formal definition of the operators, further abbreviations are useful.

Definition 5

Let \(x,y\) be cubes in I. Then \(\text{cells}(x) := \{ y \in A^I \text{ such that } (x,y) \in \text{has}_f \text{cell} \} \). We say \(x, y\) are disjoint if \(x \cap y\) and they do not share cells.

More formally \(\text{disj}(x,y) \text{ iff } x \neq y \text{ and } \text{cells}(x) \cap \text{cells}(y) = \{ \} \). In the sequel, we will use \(R\) for a role name or a feature name, and \(R^f(a)\) as shorthand for all R-fillers of a, i.e. \(R^f(a) = \{ b \in A^I \cup \text{dom}(D) \text{ such that } (a,b) \in R^f \} \).

Now to define the semantics of the operators, the intuition behind each operator is first given by an example and then defined formally. In general, so that it satisfies an assertion of the form \(x = \text{op}(y, \ldots)\), there has to exist a mapping \(\pi\) from the cells of \(x\) into the cells of \(y\). Further properties of the mapping \(\pi\) depend on the kind of operator and its parameters.
Since Multidimensional modeling is mathematically closely related to descriptive statistics Some Practical Examples solved by using mathematical operations.

Example 1
To restrict the cube battery_sales to those cells containing information about battery built in 2003 and for which at least 60 units were sold. The corresponding assertion is, battery_sales_yur = restrict(battery_sales, =2003(year) and >60(unit)).
Here ≥n stands for the unary predicate which tests for equality with n and >n stands for the unary predicate which tests for being greater than n.

1. By Definition 4
An interpretation I satisfies an assertion x = restrict(y,C) iff disj(x, y) and there exists a mapping π such that
π : cells(y) ∩ C' → cells(x) is bijective and
∀ c ∈ cells(y) ∩ C', ∀ R: R*(c) = R*(π(c)).

Example 2
The dimension year in the cube battery_sales_yur has a single value in its domain. So, we can remove this dimension without losing any information or destroying functional dependencies. The corresponding assertion is,
battery_sales_yur_03 = destroy(battery_sales_yur, year)

2. By Definition 4
An interpretation I satisfies an assertion x = destroy(y, f) iff disj(x, y), and for all z, z' ∈ cells(y) it holds f*(z) = f*(z'), and there exists a mapping π such that π : cells(y) → cells(x) is bijective and
∀ c ∈ cells(y), f*(π(c)) is undefined and ∀ R if R ≠ f then R*(c) = R*(π(c)).

Note that an arbitrary feature f can be destroyed from a cube x even if not all cells in x have the same f-fillers. In this case, by applying destroy, (1) information loss and (2) might have different cells c whose remaining f-fillers coincide.

Example 3
Suppose we have given the cube battery_sales, and another cube battery_prices containing information about a battery price with respect to its model and its construction year. If we want to enhance the battery_sales by the prices of the battery, we can do this by, battery_sales_prices = join(battery_sales, battery_prices).

3. By Definition 4
An interpretation I satisfies an assertion x = join(y, y') iff disj(y, y'), disj(x, y), disj(x, y'), and there exists a mapping π such that
π : common(y, y') → cells(x) is bijective and
∀ (c, d) ∈ common(y, y') ∀ R: R*(c) U R*(d) = R*(π(c,d)).

where common contains all pairs of cells of y and y' agreeing on all common features in y and y', i.e.,
common \( (y, y') := \{(c, d) \in \text{cells}(y) \times \text{cells}(y') \text{ such that } \forall f \in N_F: c \not\in \text{dom}(f^i) \text{ or } d \not\in \text{dom}(f^i) \text{ or } f^i(c) = f^i(d)\}.\)

where \( \text{dom}(f^i) \) is the domain of \( f^i \), i.e. \( \text{dom}(f^i) = \{x \in A^i \text{ such that } f^i(x) \text{ is defined}\}.\)

Hence for this \textbf{join} operator the features on which the cubes are joined are those which occur in both input cubes. If one wants to use \textbf{join} in a different way, one may use the operator \textbf{rename} on the input cubes to change the feature names:

An interpretation \( I \) satisfies an assertion \( x = \text{rename}(y, f, g) \) iff \( \text{disj}(x, y) \), and there exists a mapping \( \pi \) such that

\[ \pi : \text{cells}(y) \rightarrow \text{cells}(x) \text{ is bijective and } \forall c \in \text{cells}(y) \forall R: R^i(c) = R[f \rightarrow g]^i(\pi(c)), \]

where \( R[f \rightarrow g] = R \) if \( R \neq f \) and \( R[f \rightarrow g] = g \) if \( R = f \).

\textbf{Example 4}

Suppose we have the same cube battery_prices as in the previous example, but instead of just adding the prices of the battery to battery_sales, we want to see the sales volumes for each model, color and year besides the number of units sold. To enhance battery_sales by the sales volumes of the battery, we need a more powerful form of the \textbf{join} operator, namely one which takes additional functions. The above example can then be obtained by,

\[ \text{battery_sales_volumes} = \text{Join}(\text{battery_sales}, \text{battery_prices}, <volumes, mult, units, price>), \]

where \( volumes \) is a feature name that does not occur neither in \( \text{battery_sales} \) nor in \( \text{battery_prices} \). For each cell in \( \text{battery_sales_volumes} \), its \text{volumesfiller} is the product of the unit- and price-filler of the according cells in \( \text{battery_sales} \) and \( \text{battery_prices} \).

\textbf{4. By Definition 4}

An interpretation \( I \) satisfies an assertion \( x = \text{Join}(y, y', <n_1, o_1, f_1, g_1>, ..., <n_m, o_m, f_m, g_m>) \) iff \( \text{disj}(y, y'), \text{disj}(x, y), \text{disj}(x, y') \), and there exists a mapping \( \pi \) such that

\[ \pi : \text{common}(y, y') \rightarrow \text{cells}(x) \text{ is bijective and } \forall (c, d) \in \text{common}(y, y')\]

\[ \forall 1 \leq i \leq m; n_i(\pi(c, d)) = o_i(f^i(c), g^i(d)) \text{ and } \forall R \text{ if } R^i(f^i, ..., f^m, g^i, ..., g^m) \text{ then } R^i(c) \cup R^i(d) = R^i(\pi(c, d)). \]

\textbf{Example 5}

Finally, suppose we want to compute the number of units sold for each model and each year. This can be done using the operator \textbf{aggr} in the following way,

\[ \text{battery_sales_total_year} = \text{aggr}(\text{battery_sales}, \text{unit}, \text{sum}, \text{color}) \]

The operator \textbf{aggr} takes, besides the name of the cube, three parameters.

The feature unit whose fillers are aggregated using the aggregation function sum \( \in \text{agg}(D) \), and the feature color which does no longer occur in the output cube since we have summarized all cells regardless of their color-fillers.
5. By Definition 4

An interpretation I satisfies an assertion $x = \text{aggr}(y, f, S, g)$ iff $\text{disj}(x, y)$, and there exists a mapping $\pi$ such that

$\pi : \text{maxss}(y, f, g) \Rightarrow \text{cells}(x)$ is bijective and $\forall c \in \text{cells}(x): c \notin \text{dom}(g^I)$ and

$\pi'(c) = \sum \pi'(c')$ (for $c' \in \pi^{-1}(c)$) and $\forall R: \text{if } R \notin \{f, g\}$ then

$R^I(c) = R^I(c')$ for some $c' \in \pi^{-1}(c)$,

where $\text{maxss}$ contains all maximal subsets of cells of $y$ which agree on all features but $f$ and $g$, i.e.,

$\text{maxss}(y, f, g) := \{ S \subseteq \text{cells}(y) \text{ such that } \forall c, c' \in S \forall R \text{ if } R \notin \{f, g\} \text{ then } R^I(c) = R^I(c') \}$

Conclusion

- New approach for modeling multidimensional databases.
- Presentation of object-centered formalism in particular the syntax and the declarative semantics.
- The description logic ALC (D) is extended by a set of operators that work on cubes, for example for aggregating information and functional dependencies. Aggregation functions are not restricted to a fixed set, but allow for arbitrary ones.
- Distinguish between the internal representation of a cube and its visualization. Given an instance $c$ of cube as described above, $c$ can contain much more information than what can be shown in one single cube. In order to visualize the information contained in $c$, one has to choose at most 3 dimensions $d_1, d_2, d_3$ plus $n$ measures $f_i$. Then, this part of the information contained in $c$ can be visualized as cube representation provided that $<f_1, \ldots, f_n>$ is functionally dependent on $d_1, d_2, d_3$.
- The push and pull operators take a cube and transform dimensions into measures and vice versa. In this approach, this can be reduced to asking for a different visualization.
- The framework presented here is sufficiently flexible to admit with relative ease the introduction of new description constructors, which can be application specific.