DeltaMaster clicks 156

BISSANTZ

November/December 2017

Determining standard selection dynamically in relational applications

Greetings, fellow data analysts!

One of the big changes in DeltaMaster 6 is a significantly expanded backend for relational applications. These do not require a cube; instead, they make do with tables from a relational database such as Microsoft SQL Server, Oracle, SAP HANA, IBM DB2, or MySQL. This often goes unnoticed by report recipients at management level because it works largely in the same way as DeltaMaster usually does. Of course, there are some differences below the surface – not least in terms of language, as relational databases are queried using SQL rather than MDX. This has consequences for the dynamic configuration of the filter context, for example. In the last issue, we described this configuration for OLAP. This time, we are turning our attention to relational applications – and getting more technical. Our target audience is the report editors who set up such applications. That said, the result is not technical at all. Rather, it represents another component of automated reporting: individual default settings that meet users' needs.

Best regards,
Your Bissantz & Company team

Standard (default) selections can also be defined in relational applications. The effect in Presentation Mode is the same as for OLAP databases and as described in DeltaMaster clicks 155: When a report is opened for the first time, DeltaMaster runs the queries for the default selection and sets the corresponding filters in each dimension (view). This allows filters to be pre-filled automatically, including based on dynamic criteria, depending on the current date, the current user or the available data in particular.

Different languages, different structures

The differences relate to syntax but they are also conceptual. Unlike MDX databases, SQL databases have no hierarchies, no levels, and hence no standardized expressions for describing hierarchical

Online help
Directly in DeltaMaster:
menu ⓘ/*Help* or *F1* key

Support hotline
support@bissantz.com
Tel. +49 911 935536-700

DeltaMaster clicks
Archive and subscriptions:
clicks.bissantz.de

Customer portal
Latest version of DeltaMaster:
www.bissantz.de/login

Blogs
Me, myself, and BI –
Bissantz ponders
blog.bissantz.com

Bella consults – Musings of the office dog at Bissantz
www.bella-consults.com

Ready, steady, cube – Best practices from our consultants
crew.bissantz.de

Bissantz researches –
The latest from our laboratories
forschung.bissantz.de

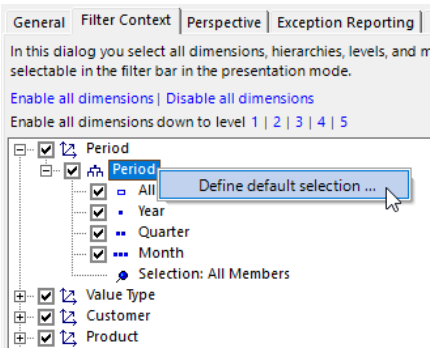
Training
We offer over 60 training sessions for DeltaMaster and Microsoft SQL Server/Analysis Services each year.
www.bissantz.com/trainings

Events
Experience DeltaMaster live – at client meetings, seminars, conventions, or trade fairs.
www.bissantz.com/events

structures. However, these are precisely what management information needs in order to derive findings from data! DeltaMaster closes this gap by offering mechanisms for realizing multidimensional ways of thinking and working even when using relational databases. This is achieved using SQL and a few additional rules, according to which DeltaMaster transfers the results of a relational query into the desired multidimensional constructs.

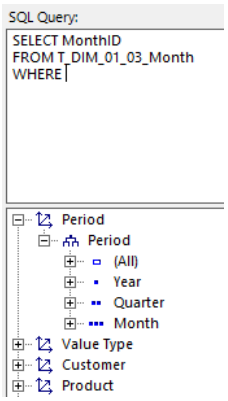
Report Properties, Filter Context

Default selection forms part of the *Filter Context*. This can be edited in its own tab in the *Report Properties* (context menu of reports in Modification Mode) in the same way as for MDX databases. You can define the *Default Selection* in the hierarchy context menu.

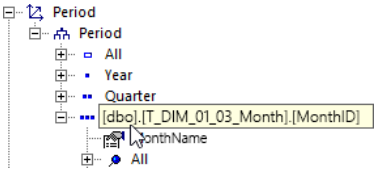


This opens the familiar *SQL Editor*. Enter the query in the upper section of the dialog. The lower section shows all of the dimensions, hierarchies, levels and members as well as the measures.

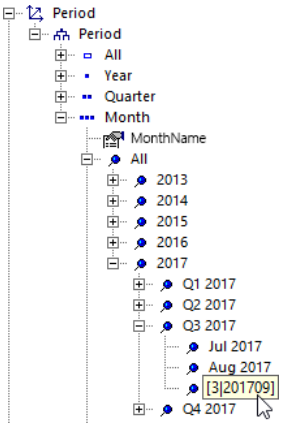
If you move the mouse pointer to an item while holding down the *Alt* key, Delta-Master displays technical information as a tool tip. This is required in SQL queries and can be added to the input field by dragging and dropping or double-clicking.



For levels, DeltaMaster displays the qualified name of the key field. This comprises the schema (in the illustration: dbo), the table name (T_DIM_01_03_Month) and the field name (MonthID).



For members, DeltaMaster displays the ordinal number of the hierarchical level, i.e. essentially its depth, and the key. In the illustration, the month is level 3 (the top level is always 0) and the key is 201709.



SQL queries

The way in which queries are formulated for the default selection is based on the respective dimension. We make a distinction between

- regular level-based dimensions generated from the columns of one or more tables and
- virtual time dimensions based on a date field, e.g. data types such as date or date time.

The virtual time dimension is one of DeltaMaster's specialties. In relational applications, it removes the need for some of the typical steps in data preparation: Instead of distributing stored dates across separate columns for days, months, quarters, years, etc. as would be required in a regular hierarchy, DeltaMaster can process the data field directly and construct a corresponding virtual hierarchy independently in a similar way as for the so-called server time dimension of Analysis Services.

For both types of dimension, the query result depends on the first column only; additional columns are ignored. This first column contains the values to be interpreted as the member key in the form of strings (e.g. data types such as char or varchar) or numbers (e.g. integer).

In most use cases, the default selection should be exactly one member, not multiple members. Accordingly, the query must return exactly one value (one column, one row). To ensure that a query returns exactly one row, and hence that exactly one member is selected, you can restrict the number of result rows (in Microsoft SQL, for example, using a query like “SELECT TOP 1 FROM …”) if this does not already follow from the query logic, e.g. the WHERE clause.

If multiple rows are returned, DeltaMaster treats this as a multi selection. In this case, the members must belong to the same dimensional level; different levels are not permitted. Duplicate rows can be prevented in Microsoft SQL Server using the suffix DISTINCT (“SELECT DISTINCT <Field> FROM …”).

Regular dimensions

One important aspect of syntax in regular dimensions is whether a hierarchical level is specified. In turn, this depends on how many levels the dimension has.

- In dimensions with one or two levels, you do not need to specify the level.

  If a dimension has exactly one level, DeltaMaster looks for the member at that level. Examples include value types (target, actual, …) and currencies. In these cases, selecting the member key is sufficient:

  SELECT 'P' – “Target” member in the value type dimension

A default selection could be defined in the value type dimension; "P" may represent the key for the "Target" (i.e. the plan) member. In practice, of course, you would not select a constant member but would determine the member dynamically (see following section); constant members are used here for illustrative purposes only.

If a dimension has two levels, i.e. a top level and a primary level, DeltaMaster automatically looks for the member in the primary level; so that it is not necessary to specify the level. In this case, too, selecting the key is therefore sufficient. In our reference use case, "Chair", colors and sales teams are examples of such dimensions.

- In dimensions with more than two levels, the level must be specified.

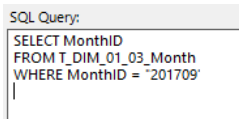
  This means assigning an alias in the query or specifying the key field:

  ```
  SELECT '201709' AS MonthID
  ```

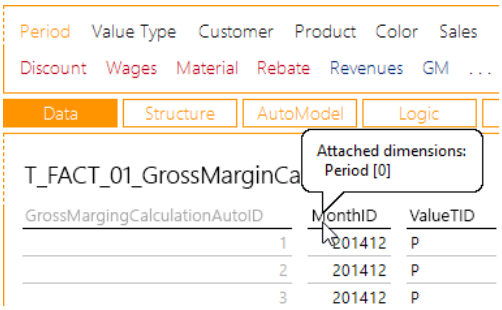
  or:

  ```
  SELECT MonthID
  FROM T_DIM_01_03_Month
  WHERE MonthID = '201709'
  ```

  The name of the key field (in both examples: MonthID) and the table name (in the second example: T_DIM_01_03_Month) are determined in *SQL Editor* as described above (*Alt*+mouse pointer, drag and drop or double-click to add).



When formulating a query, it can be useful to take a look at the data. The tables and their key fields and characteristics can be viewed on the *Data* page in *Modeling*. The table name in caps (T_FACT_01_GrossMarginCalculation) also acts as a menu for switching to the other tables. A tool tip in the column header shows how the column is used (*Alt*+mouse pointer).



Date, user, measure

In the above examples, we have specified constant keys in order to keep things simple. However, the very purpose of default selection is to determine the key dynamically depending on changing conditions. The most frequent examples are the system date and the current user name. In Microsoft SQL Server, this information can be obtained using the GETDATE() and SYSTEM_USER functions; similar functions are available in other databases. The returned values must be transformed into strings that match the member keys in the respective dimension.

Examples:

```
SELECT CAST(YEAR(GETDATE()) AS VARCHAR) +
RIGHT('0' + CAST(MONTH(GETDATE()) AS VARCHAR), 2)
AS MonthID -- for period descriptors such as "201709" in string form

SELECT SYSTEM_USER AS UserID -- including domain
```

```
SELECT SUBSTRING(SYSTEM_USER,
CHARINDEX('\', SYSTEM_USER) + 1, LEN(SYSTEM_USER)) AS UserID -- excluding domain
```

The last two examples require users to be modeled as a separate dimension so that the desired member can be taken over directly from the name of the system user. Often, however, the user name needs to be allocated, e.g. to the sales team, product group or customer group. In simple cases with few allocations, these can be recorded in the default selection, e.g. as follows:

```
SELECT CASE
WHEN SYSTEM_USER = 'chair\baumann' THEN 'V2'
WHEN SYSTEM_USER = 'chair\hohlmaier' THEN 'V1'
END
AS SalesID
```

For more extensive use cases, it is advisable to store such allocations in the database table.

The default selection can also be made dynamic by including measures. The following query returns the last month for which (positive) actual revenue is already recorded. This means the default selection in the period dimension continuously adjusts to reflect the transaction data.

```
SELECT TOP 1 MonthID
FROM T_FACT_01_GrossMarginAccounting
WHERE ValueTypeID = 'I' AND Revenue > 0
ORDER BY MonthID DESC
```

In this query, DeltaMaster accesses the fact table. This can be viewed in *Modeling* as shown above.

Virtual time dimension

For virtual time dimensions, DeltaMaster always requires the desired level (time granularity) to be specified, e.g. day or month. As this cannot be expressed using SQL, DeltaMaster supports its own syntax: The desired level is specified by adding a DeltaMaster-specific code before the date in text format, followed by an apostrophe as a delimiter. Example:

```
SELECT '3''04/28/2017 00:00:00'
```

This sets April 2017 as the default selection in the period dimension based on the date April 28, 2017. Within DeltaMaster, dates are recorded using an invariant culture that is not region-specific or user-specific. The US date format (“MM/DD/YYYY”) is used as illustrated by the example. The first and last apostrophe are the string delimiters in SQL. Code 3 indicates that the desired level of time granularity is the month. The apostrophe after the code is duplicated because it falls within the delimiters and hence must be masked (escape).

DeltaMaster recognizes the following codes for time granularity:

Code	Level
0	Year
2	Quarter
3	Month
4	Week
5	Day

In practice, of course, codes are not combined with a fixed date but with a dynamic SQL function like GETDATE() that selects the current date or the computer's system time. The corresponding queries take the following form:

```
SELECT '5''' + CONVERT(CHAR(10), GETDATE(), 101) + ' 00:00:00' -- Select day

SELECT '3''' + CONVERT(CHAR(10), GETDATE(), 101) + ' 00:00:00' -- Select month
```

The internal structure of the virtual time dimension means the date can be easily derived from the system time, so the query is shorter than in the example shown in the previous section. Instead of assembling strings, the system time can be converted into the required format using the CONVERT function; in Microsoft SQL Server, the code for this is 101. In order to make the query independent of the time of day, only the first ten digits of the system time are taken into account, i.e. the date, and a "neutral" time of day is added.

An alias, field descriptors, table names, or similar are not required, as the virtual time dimension is not based on different database fields.