Data Model Editor : Overview

Top Next

Data Model Editor is intended for creation database description used by Easy Query controls. This description allows to hide the details of your database organization from the end-user. It contains information about tables, links between them, some entities of the data domain. In addition this description includes list of operators used in conditions (such as "is equal to" or "is more than") and list of custom (manually defined) conditions which can be used in queries.
Though being easy to use and self-explanatory, data model editing requires some understanding of the relational databases handling. Users totally unfamiliar with database management should better leave this dialog for an database operator or another experienced user.

This dialog contains such pages:
- Tables - describing tables taking part in the data model;
- Entities - describing the entities and their attributes which end-user can operate with to build queries;
- Operators - describing operations (like comparisons) upon the data model's fields;

Each of the pages is described in detail in appropriate topics of this help.

Use *Load* and *Save* speed buttons to load or save the data model from/to XML file.

Data Model Editor: Tables and links page

Previous Top Next

Use this page to describe tables taking part in the data model.
The page consists of 2 main parts: list of tables in the left side and properties of selected table at the right (the second panel is empty when no table is selected).
The table properties panel are also diveded on 2 panels: the properties themselves on the top and the list of table links at the bottom.

To select the table or link just click on it. Properties of the selected table will be shown in the right panel.
To add or delete tables use the corresponding menu items from "Tables" menu.

Table properties:

Table alias is an alias for the table in generated SQL statements (optional but needed if you link the table more than once on different conditions)
NOTE: any particular database table can be added to the data model several times with different aliases. It is necessary to eliminate the ambiguities when some two tables have several link paths from one to another. For more information see: Working with aliases.

Table hint(s) is intended to specify locking method used in MS SQL Server syntax (like NOLOCK, ROWLOCK or READCOMMITED);

Quote table name check box means if the table name should be put in double quotes in SQL statements - it is useful for table names including spaces and national characters.

Working with table links
To add a new link for selected table just click on Add button near the list of links. Speify link properties

and conditions using Edit link dialog and press OK. New link will apear in the list.
To edit existing link - just select in the list and click on Edit button. Press OK when finish editing.
To remove some link - selected it and press Delete button.

Working with aliases

Previous Top Next

When do you need aliases? Sometimes you want to use one table for several different links and only your users query can define which link you want to use to build correct query.
To build such links you need to use aliases. Let's see the following example.

You want to have links like Table1.FieldA -> Table2.FieldC and Table1.FieldB -> Table2.FieldC
You have a database that contains the information about people migration within the United States. There are tables **People** and **States** in your database. **StateFrom** field contains 2 letter state abbreviation of the state where the person from, and StateLives contains the state abbreviation of the state where the person lives now. State table contains StateID - this 2 letter key and StateName - long state name.

People	States
PersonID	**StateId**
Name	StateName
StateFrom	
StateLives	
Phone	

Open Data Model Editor and add People table. After this add States table twice and set Table Aliases to 'StateFrom' for first table and 'StateLives' for second. Add 2 different links to table People:
1. People.StateFrom -> StateFrom.SateId
2. People.SateLives -> StateLives.SateId

Then go to the Fields page and create two fields:
1. "State person from" which corresponds to StateName field from the States table with StateFrom alias
2. "State person lives now" - corresponds to StateName field from States table with StateLives alias.

Now your users can create queries and use 2 States tables which a really connected to one table in the database. SQL standard allows you to create such queries when you use aliases and Simple Query does too. See the example on the picture:

Here we ask for the people who have arrived from California and do not live in Oregon now.

Data Model Editor: Edit link dialog

Previous Top Next

In this dialog, you can define the following properties for some link between tables:

- **Table1** and **Table2** combo box - can be used to choose the linked tables.
- **Join type** group of radio buttons allows you to specify the type of join (inner, left outer, right outer or full join).
- **Join conditions** panel - allows you to specify the conditions of the link.

The "Join conditions" panels consists of two main parts. Firs part (at the top) is the list of conditions, second (at the bottom) contains different control which
allows to define new condition and add it into the list.
Delete button at the right of condition list deletes a selected condition. *Clear* button - clears all conditions completely.

To add new condition you need to do the following:
1. Select the type of each condition part (left and right) ;
2. Select table field (in case of "Field" type) or enter a constant expression (for "Constant" type) for each condition part;
3. Sselect operator ("=" by default);
4. Press "Add Condition" button.

Data Model Editor: Entities page

Previous Top Next

This page describes entities and participating in the data model and their attributes.

Entities are intended to hide data storage details from the end user. Instead of operating with tables, views and fields (such as Orders.CustNo, Customers.Addr1) user see some entities from real world (Order, Customer, Vendor, etc.) and their attributes (Customer Name, Order Ship Date, Vendor Country,

etc.).
For large data models entities can be organized in hierarhy for more easy manipulation.
There are two main types of entity attributes:

- Data Attribute - it is totally corresponds to particular field in some database table (e.g. attribute Order Ship Date corresponds to database field Orders.ShipDate);
- Virtual Attribute - a calculated attribute which is defined by some expression containing several fields, operators (+, -, ||, etc.), constants and even functions or storage procedure calls;

*E*ntities tree holds list of all defined entities and their attributes.

Main operations:

- To add a data attribute - select the entity node you would like to add it to and then choose *Add Data Attribute* item from "Entities" menu. Select the table, its field in the dialog that appear and click *OK* button.
- To add a virtual (i.e., calculated) attribute, select the entity node you would like to add it to and choose *"Entities | Add Virtual Attribute"* menu item*;*
- To delete an attribute or entity - select it and choose *"Entities | Delete selected item"* menu item;
- To edit an attribute, select it in the tree and change its properties in the editor appearing in the dialog's right part.
- To move the attribute from one entity to another (or to change its appearance order within the entity) - just drag it to the appropriate place.

The attribute's property editor has *General*, *Operators* and *Value Editors* tabs described below.

***General* tab**

This tab contains the following input field and checkboxes:

Caption field - a string meaning how the attribute name will be displayed for an end-user;

Expression field - contains table and field names for data attributes and full SQL expression for virtual ones. For data attributes this field is read-only.

Used tables panel - contains the list of all tables used in the selected attribute. For data attributes this field is read-only.

Data Type and **Data Size** fields - contain type of attribute and its size. These values are taken from corresponding field definition for data attributes and should be set manually by data model developer for virtual attributes. See Data types for details.

Use in conditions check box allows to specify if this attribute can be used in query conditions. If this option is not checked - user will not see this attribute in drop-down list in query panel;

Use in result check box allows to specify if this attribute can be used in result columns. If this option is not checked - user will not see this attribute in drop-down list in columns panel;

Use in sorting check box allows to specify if the user can sort by this attribute.

Quote field name in SQL check box means if the field name should be put in quotes in SQL statements. Useful for field names which includes some reserved words. Field names with spaces will be quoted automatically.

Use alias check box specifies if query builder will generate alias for the result column where this attribute is used.

Description field - allows to associate some textual description with attribute. Entered data can be accessed at run-time.

Custom Data field - allows to associate any information with attribute. Entered data can be accessed at run-time.

***Operators* tab**

This tab holds list of all operations applicable to the attribute. Add or remove operations by corresponding buttons. Use *Clear* button to remove all operators from the list.
Use *Defaults* button to reset the list of operators for selected attribute to default state. The default operator list contains most appropriate operators for attriubte's data type.
See Operators page for details about operators used in data models.

***Value Editors* tab**

This tab allows to define how user will edit the parameters which the selected attribute is compared to in query conditions. To learn more about available editors see Value Editors topic. To specify editor parameters click on "Settings..." button.

Data Model Editor: Operators page

Previous Top Next

This page defines operators which can be used in conditions (such as 'is equal to', 'less than' and others). List in the left part shows defined operations. Add or delete one by right-clicking the list and choosing appropriate topic from the pop up menu or using corresponding speed buttons on the top of the page. To edit an operation choose it from the list and modify its properties in the right part of the page.

Operator properties:

Operator ID is internal identifier for the operator.

Caption property allows to specify how this operator will be presented to user when he/she selects it in condition;

Display Format is a template which describes how operator (an all condition) will be shown to end user in query panel. The most usual value of this field is:
`{expr1} [[operator text]] {expr2}`
here `{expr1}` and `{expr2}` will be substituted by corresponding expression in condition (an entity attribute or constant).
The text in `[[ ]]` brackets will be shown as link for operator selection.
For "between" operator Display Format property will also contain 'and' word between second and third expression:
`{expr1} [[is between]] {expr2} and {expr3}`

Expression is a template for condition expression in generated SQL query. It may contain any correct SQL expressions (operators such as =, >,<, functions or even names of stored procedures) and the following special variables:
{expr1} - is substituted with the entity attribute selected by user;
{expr2}, {expr3}, ... - are substituted by constant values typed or selected by user or entity attributes selected at the right part of condition;

Additionally you can use the following constructions:
{exprN.table} - will be replaced by table name of N expression if this expression represents an entity attribute or by empty string in other case.
{exprN.field} - will be replaced by field name of N expression if this expression represents an entity attribute or by empty string in other case.

Examples:
For the simple "is equal to" operator the format string is:
```
{expr1} = {expr2}
```

The "starts with" operator has the following format:
```
{expr1} LIKE {expr2}
```

You can define more complex operators even with sub-queries. For example here is an expression for "more then average" operator:
```
{expr1} > (SELECT AVG({expr2.field}) FROM {expr2.table})
```

Default data type - represents expected type of operator parameters used in condition expression. "Auto" value means the same type as the type of entity attribute used at the left part of condition, other values are correspond to particular data types.

Default value kind - defines the kind of data which has the parameters of this operator.
"Scalar" - means simple single value is needed: one string, one number, etc. This kind allows also to specify entity attribute.
"Const" - the same as previous one but allows to specify only constant value (entity attributes are not allowed);
"Attribute" - the same as the first one but allows to specify only entity attribute (constant values are not allowed);
"List" type requires list of scalar values sepated by comma. E.g., having this option checked, when the user enters **a, b, c** as parameter value, it's treated as **'a', 'b', 'c'** instead of **'a, b, c'** in the generated SQL text.
"Query" type means that operator requires SQL SELECT statement as value in the right part of condition. To build this statement query panel opens seperate dialog.

Group - allows to select the group which operators belowngs to.

Default value editor - allows to specify the value ditor for operator's parameters. To learn more about available editors see Value Editors topic. To specify editor parameters click on "Settings..." button. Operator value editor has higher priority in a condition than the editor defined for entity attribute used at the left part of this condition.

Apply to types is list of check boxes defining data types to which the operation is applicable.

Main operations

To add new operator use "Operators | Add operator" menu item. To delete - use "Operators | Delete selected".

Operators menu also contains "Add/Update default operators" menu item which can be used to add the default operators in the model or update the parameters of existing one.

There are the following default operators:

ID	Caption	SQL expression
Equal	is equal to	{expr1} = {expr2}

NotEqual	is not equal to	{expr1} <> {expr2}
LessThan	is less than	{expr1} < {expr2}
LessOrEqual	is less than or equal to	{expr1} <= {expr2}
GreaterThan	greater than	{expr1} >{expr2}
GreaterOrEqual	greater than or equal to	{expr1} >= {expr2}
IsNull	is null	{expr1} is null
InList	is in list	{expr1} in ({expr2})
StartsWith	starts with	{expr1} like {expr2}
NotStartsWith	does not start with	not({expr1} like {expr2})
Contains	contains	{expr1} like {expr2}
NotContains	does not contain	not({expr1} like {expr2})
Between	is between	{expr1} between {expr2} and {expr3}
InSubQuery	is in set	{expr1} in ({expr2})
DateEqualSpecial	is	{expr1} = {expr2}
DateEqualPrecise	is	{expr1} = {expr2}
DateBeforeSpecial	is before (special date)	{expr1} < {expr2}
DateBeforePrecise	is before (precise date)	{expr1} < {expr2}
DateAfterSpecial	is after (special date)	{expr1} > {expr2}
DateAfterSpecial	is after (precise date)	{expr1} > {expr2}

"Special" and "Precise" operators with the same expressions differ by value editors. "Special" operators has "Custom List" value editor with "SpecDateValues" name. This list is filled at runtime by such values as Today, Yesterday, First day of month, etc.
The "Precise" operator has usual "Date/time value editor" which allows simply to enter necessary date (or select it using calendar).

Value Editors

Previous Top

You can select one of the following editors:

- **Auto:** means that the most appropriate value editor will be used depending on entity attribute's data type and operator which used in condition. For example for date attribute - DataTime Picker control will be shown, for boolean one - the user will get an ability to select the value from the list of two items: False and True;
- **Text editor:** the values will be edited in usual text box field. You can specify default value and type of edited data for this editor;
- **Date/time editor:** query panel will sho date/time picker control for elements with such value editor.
- **List of constants**: user will be prompted to select one (or more in case of using "is in list" operator) value from the list of available values. *Values* column stands for values accepted as the choice result (and actually inserted into SQL text); *Text* are corresponding text items displayed to user in the pop-up list.
- **Custom list**: the same behaviour as in previous case but this editor does not allow you to enter the list itself. Insted you can specify the name for this list and fill it at-runtime in RequestList event handler of QueryPanel control;.
- **SQL list**: Similar to previous two options, but the list of available values is defined by some SQL query to database. First column of the resulting record set is interpreted as actual value which will be used in result SQL statement and the second one is used as captions (the text shown to user).
- **Custom**: The programmer should provide the value editor for this field manually using special event in QueryPanel component. There is an auxiliary text field "Custom Data" where developer can enter any textual information and get access to it at run-time through CustomData property of any condition object;