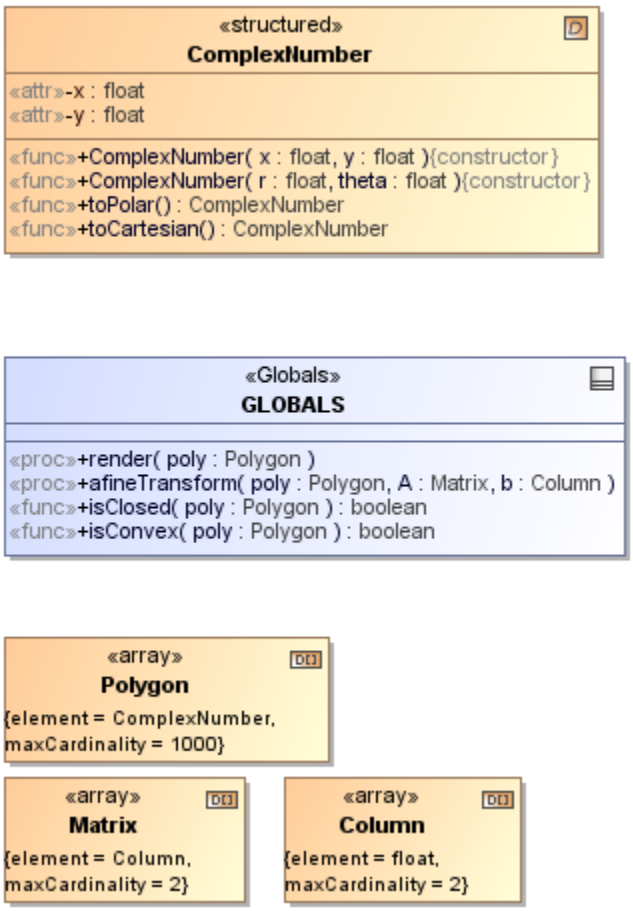
Routines

On this page

- Procedure
- Function
- Method
- Parameter
- Cursor and routine result table



Routines example.

SQL supports several different kinds of routines. There are global routines, that are not bound to a particular type but belongs to the schema. There are two kinds of these routines - Procedures and Functions. And there are routines, that are bound to a particular structured user defined type - Methods. Each routine kind can have several parameters. Parameters have type and direction (**in**, **out**, **inout**). Functions and methods in SQL have return types - this is formalized in UML models by having an additional parameter with **return** direction kind.

There is an UML limitation, that UML does not allow to place UML operations (which are used to model SQL procedures and functions) directly in the UML packages (which are used to model SQL schemas). For this reason global routines are placed into a special container class - GLOBALS (see GLOBALS).

Routines can be external (written in some other languages and attached to database engine) or SQL routines. In the latter case, body of the routine can be specified in the model. Due to UML specifics, there are two ways to specify the routine body - by filling the UML **method** field or by filling the UML **bodyCondition** field of the operation. These two ways are visible in the Specification window under the field names **Source (as method)** and **Source (as body condition)**. When specifying routine body, specify only one of these fields.

To use “as method” way

---

1. Right-click the GLOBALS element in the Containment tree and from its shortcut menu select **New Element** > **Source**. A source element (a UML OpaqueBehavior with the «Source» stereotype applied) under the GLOBALS element will be created in your schema.
2. In the Specification window of routine, edit the **Source (as method)** property value and in the opened dialog select the source element you’ve just created.

To use “as body condition” way, you simply have to fill the field. The routine body model element (in this case, UML Constraint - holding UML OpaqueExpression) shall be created under your routine model element.

Besides the standard SQL element properties, all 3 kinds of routines have the following properties available in the Specification window.

Property name	Description
Specific Name	Additional name for the routine, uniquely identifying it throughout the system.
Deterministic	Specifies whether routine is deterministic (always gives the same output with the same data input).
Parameter Style	SQL or GENERAL.
SQL Data Access	Specifies how routine accesses SQL data (NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL DATA).
Source (as method)	Fields holding routine body text (choose only one).
Source (as body condition)	
Creation TS	Routine creation and last edit timestamps.
Last Altered TS	
Authorization ID	Authorization identifier which owns this routine (owner of the schema at routine creation time).
Security	Determines the authorization identifier under which this routine runs. Typically set to “INVOKER”, “DEFINER”, “IMPLEMENTATION DEPENDENT”.
External Name	The name of the external language routine implementation method (if routine is non-SQL routine).

Procedure

> **Note**
> SQL Procedure is modeled as UML Operation with «Procedure» stereotype applied. For the sake of compactness, procedures are displayed with the «proc» keyword (instead of the long form - «Procedure») on the diagram.

Procedure is an operation that can be SQL-invoked and performs some actions depending on the parameters supplied. Procedures are global for schema - they are created under the GLOBALS model element.

Besides the standard properties of SQL routines (see Routines), procedure has the following properties available in the Specification window.

Property name	Description
Max Result Sets	If result set count is returned by procedure is dynamic, this value limits count thereof (DYMANIC RESULT SETS <max> clause).
Old Save Point	Savepoint level indicator for procedure (false means that new savepoint must be established before the procedure is run.).

Function

> **Note**
> SQL Function is modeled as UML Operation with «Function» or «BuiltInFunction» or «UserDefinedFunction» stereotype applied. By default the «UserDefinedFunction» is used, however if another kind can be freely used if it is necessary for modeling needs (e.g. if we are modeling some built in library and want to specify that functions are built in and not user defined).

Function describes some operation that calculates and returns some value depending on the parameters supplied. Functions are global for schema - they are created under the GLOBALS model element.

For the sake of compactness, functions are displayed with the «func» keyword (instead of the long form) on the diagram.

Besides the standard properties of SQL routines (see Routines), function has the following properties available in the Specification window.

Property name	Description
Null Call	Specifies that function returns NULL when called with NULL parameter value (RETURNS NULL ON NULL INPUT clause).
Type Preserving	Specifies that function does not change the type of the supplied parameter (returns the same object).
Transform Group	Allows to specify TRANSFORM GROUP <groups> clause - single or multiple.

Method

> **Note**
> SQL Method is modeled as UML Operation with «Method» stereotype applied. For the sake of compactness, methods are displayed with the «func» keyword (instead of the long form - «Method») on the diagram.

Method is a function of the structured user defined type. It is created inside the structured UDT.

Besides the properties of SQL functions (see Routines), method has the following properties available in the Specification window.

Property name	Description
Constructor	Specifies that function is a constructor (used to construct values of the enclosing structured UDT).
Overriding	Specifies that function is overriding the same-named function from the parent structured UDT.

Parameter

> **Note**
> SQL Parameter is modeled as UML Parameter with «Parameter» stereotype applied.

This model element specifies data inputs / outputs into routine calculations. Parameter has a type, direction (in / out / inout for usual parameters and a single parameter with direction return for functions) and default value.

Besides the standard SQL element properties, parameter has the following properties available in the Specification window.

Property name	Description
Type	Type of the parameter.
Type Modifier	
Default Value	Default value (used when value is not supplied during routine invocation).
Direction	Direction of data flow through the parameter (into the routine, out of the routine or both).
Locator	AS LOCATOR modifier of the type. Specifies that instead of value, means to locate value are transferred.
String Type Option	Only valid when parameter type is XML type. Specifies underlying string datatype.
Cast Type	Additional options, specifying that return parameter is cast from another type (possibly with locator indication).
Cast Locator	

Cursor and routine result table

> **Note**
> SQL Cursor is modeled as UML Parameter with the «Cursor» stereotype applied.

When routine does not return a scalar value but a collection of the table values, cursor is used, instead of the parameter. Cursor has a type. This type must be some table type instead of the scalar types used for parameters. It can be an actual table / view from the model, if the cursor returns values from that table, or (if cursor returns data from some SELECT statement) can be a synthetic table. A Routine Result Table model element is used for this purpose (UML Class with «RoutineResultTable» stereotype applied). Its modeling is exactly the same as the normal tables - this is just an ephemeral table.

Besides the standard SQL element properties, cursor has the following properties available in the Specification window.

Property name	Description
Type	Type of the cursor. Should point to the routine result table.
Type Modifier	
Direction	Direction of data flow through the parameter (into the routine, out of the routine, routine result).