Constraints

On this page

Tables have a multitude of constraints between them. These constraints enforce the proper business semantics for the data in the database tables (relationships between data in different tables, semantical constraints of the data in the table). These available constraint types are as follows.

- Primary key constraints - specifying column (or a combination of columns), which uniquely identify the row in the table.
- Unique constraints. They are very similar to primary key constraints - uniquely identify the row in the table. One of the unique constraints of the table is designated as primary.
- Foreign key constraints, which establish relationships between two tables.
- Nullability constraints (NOT NULL constraint) - a simple constraint on the column, indicating that column must have value
- Check constraints establish additional checking conditions on values in the column / table.
- Assertions provide more global check than a check constraint - spanning multiple tables.
- Indexes are not constraints per se, but they are covered in this section because they are modeled similarly.

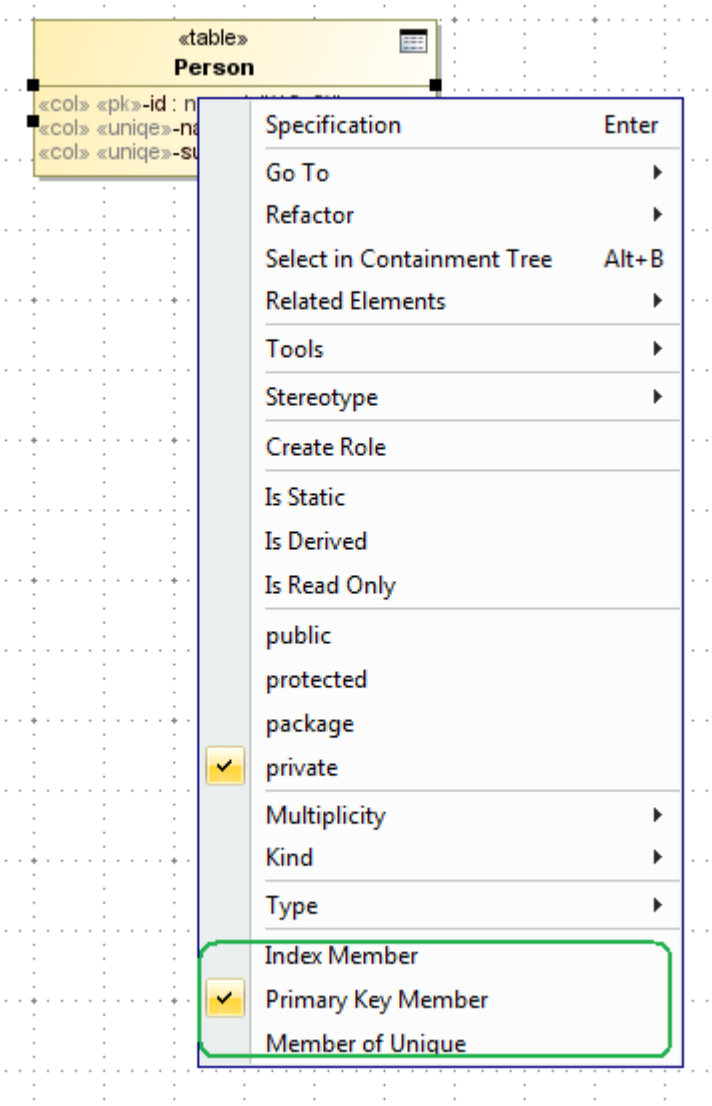
The primary keys, unique and check constraints, indexes can be modeled in two ways. One way is easy and simple but does not cover all the options provided by SQL. Another way is tedious, but provides full SQL coverage.

Implicit primary key, unique, check constraint, and index modeling

Note

- SQL Primary Key (when implicitly modeled) is modeled as an additional «PrimaryKeyMember» stereotype applied on the SQL column. This variant is shown in the diagram as an additional «pk» keyword on the column in the diagram.
- SQL Unique Constraint (when implicitly modeled) is modeled as an additional «UniqueMember» stereotype applied on the SQL column. This variant is shown in the diagram as an additional «unique» keyword on the column in the diagram.
- SQL Check Constraint (when implicitly modeled) is modeled as an additional «CheckMember» stereotype applied on the SQL column. This variant is shown in the diagram as an additional «chk» keyword on the column in the diagram.
- SQL Index (when implicitly modeled) is modeled as an additional «IndexMember» stereotype applied on the SQL column. This variant is shown in the diagram as an additional «idx» keyword on the column in the diagram.

An easy way of modeling this kind of constraint is applying the «PrimaryKeyMember», «UniqueMember», «CheckMember», or «IndexMember» stereotype on the necessary column. PK, unique, and index markers can be applied on the column via its shortcut menu, as shown in the following figure.



Quick application of PK, Unique, and Index markers.

To apply a check constraint marker on a column

---

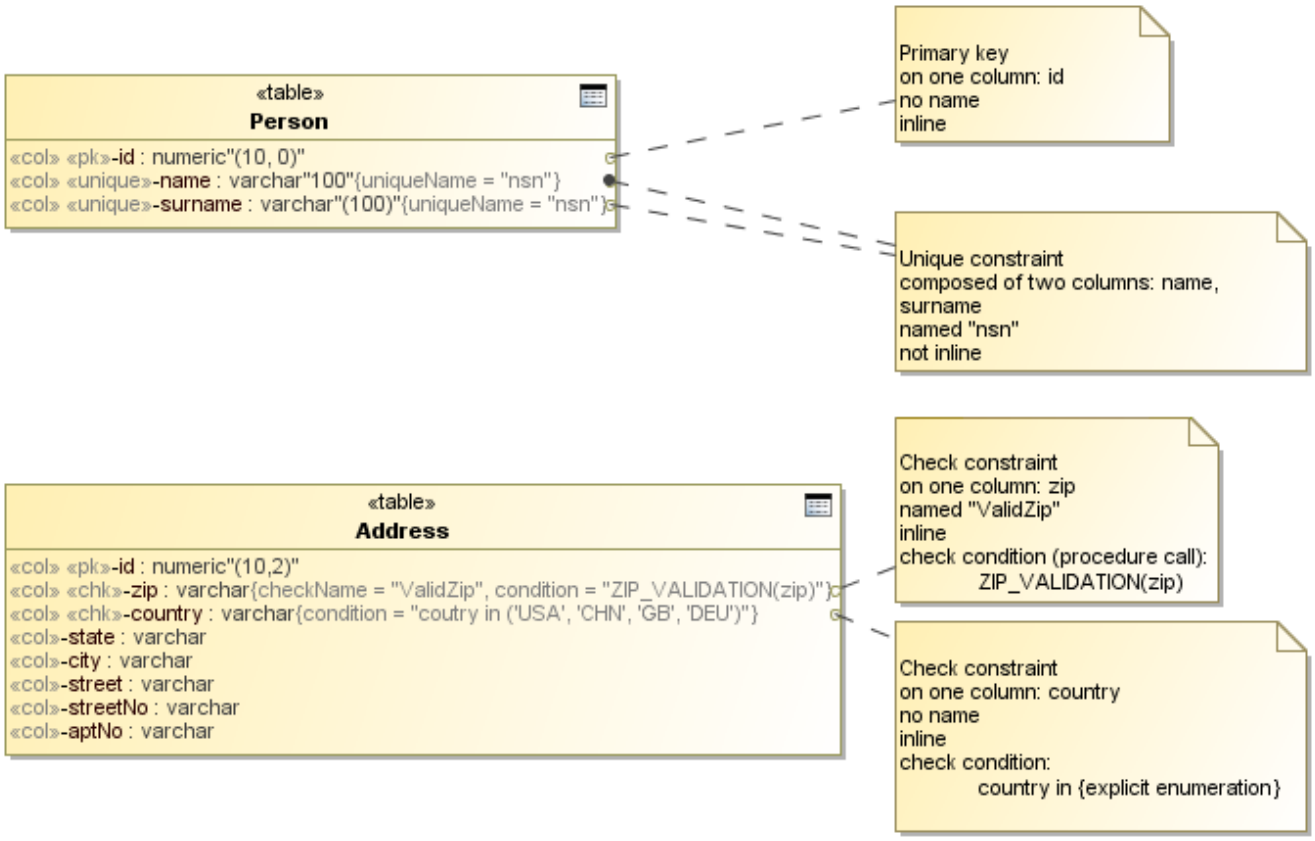
1. Open the Specification window of the column.
2. Define the Condition property value in the In Check Constraint property group.

Thusly marked column is considered as a member of one-column constraint, specified in-line. It is by default an unnamed constraint. To specify its name, you need to define the **Primary Key Name**, the **Unique Key Names**, the **Check Name**, or the **Index Names** property value in the column Specification window.

In the SQL script (in CREATE TABLE, ADD COLUMN statements) this would correspond to the following part of the column specification.

```
<column name> [ <data type> ] ...
[ [<constraint name>] <constraint>... ]
<constraint> ::=
| UNIQUE| PRIMARY KEY
| CHECK '('<condition>')'
```

If primary key, unique constraint or index must span several columns (in this case constraint is not in-line, near the column definition, but as a separate definition at the bottom of the table definition), all the columns must be marked with the appropriate «UniqueMember» / «IndexMember» stereotype and all must have the same name. Column can participate in several unique. Various cases of quick constraint modeling are depicted in the following figure.



Various situations, modeled with quick constraint notation.

Explicit primary key, unique, check constraint, and index modeling

> **Note**
>
> - SQL Unique Constraint (when explicitly modeled) is modeled as UML Constraint with «UniqueConstraint» stereotype applied.
> - SQL Check Constraint (when explicitly modeled) is modeled as UML Constraint with «CheckConstraint» stereotype applied.
> - SQL Index (when explicitly modeled) is modeled as UML Constraint with «Index» stereotype applied.

The quick, implicit way to model constraints does not cover some cases, allowed by SQL. Constraints in SQL can be marked as DEFERABLE, INITIALY DEFERRED, constraint in the database can be in active state (enforced) or disabled. Indexes have various configuration parameters.

Modeling with the help of «XYZMember» stereotypes does not allow to specify this additional information. In this case modeling with explicit constraint model elements is necessary. This can be done from the Specification window of table. There are separate tabs for creating these constraint elements: **Unique Constrains** (allows creating both primary keys and unique constraints), **Check Constraints**, **Indices**. Once created, additional properties of the constraints can be specified.

Besides the standard SQL element properties, primary key and unique constraint have following properties

Property name	Description
Members	Columns, constrained by this constraint (must come from the same table where constraint is located.)
Inline	Whether constraint is defined near the column definition, or separately, at the bottom of the bale definition. Only one-column constraints can be inline.
Deferable	Marks the constraint as deferrable.
Initially Deferred	Marks the constraint as initially deferred.
Enforced	Whether constraint is actively checked in the database (can be changed with the SQL statements).

Check constraints have the same properties as primary key and unique constraints, and additionally have following properties available in the Specification window.

Property name	Description

Condition	Condition to be checked

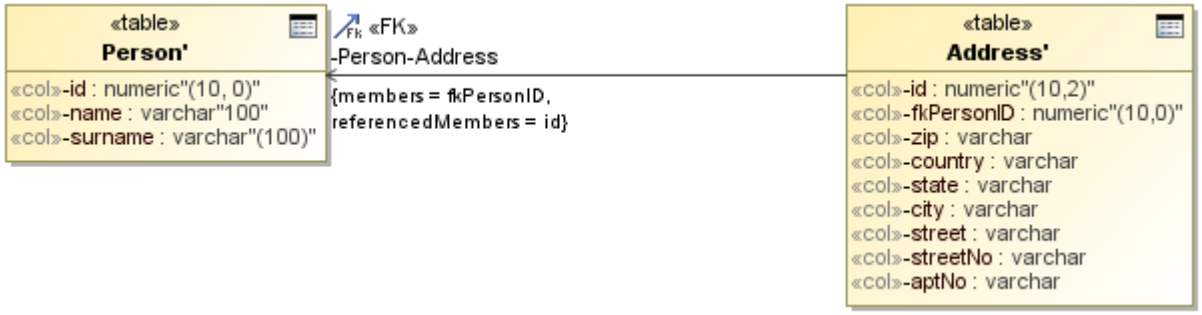
Besides the standard SQL element properties, index has the following properties available in the Specification window.

Property name	Description
Members	Member columns of this index
Member Increment Type	For each member column, ASC or DESC ordering direction
Unique	Index is used to enforce uniqueness
System Generated	Index is system-generated.
Clustered	The index is clustered. Only one index per table can be clustered. Non- clustered indexes are stored separately and do not affect layout of the data in the table. Clustered index governs the table data layout (table data pages are leafs of the index tree).
Fill Factor	Hash table fill factor of the index
Included Members	Additional member columns of the index. No sorting is done by these columns, however their data is included into index - this provides fast retrieval. This feature is very database-specific (AFAIK only MS SQL Server has those).
Included Member Increment Type	

Foreign keys

Note

- SQL Foreign Key (when modeled with UML Association relationship) is modeled as UML Association with the «FK» stereotype applied on the end of the association (UML Property), which points to the referenced table
- SQL Foreign Key (when modeled with UML Constraint) is modeled as UML Constraint with «ForeignKey» stereotype applied.



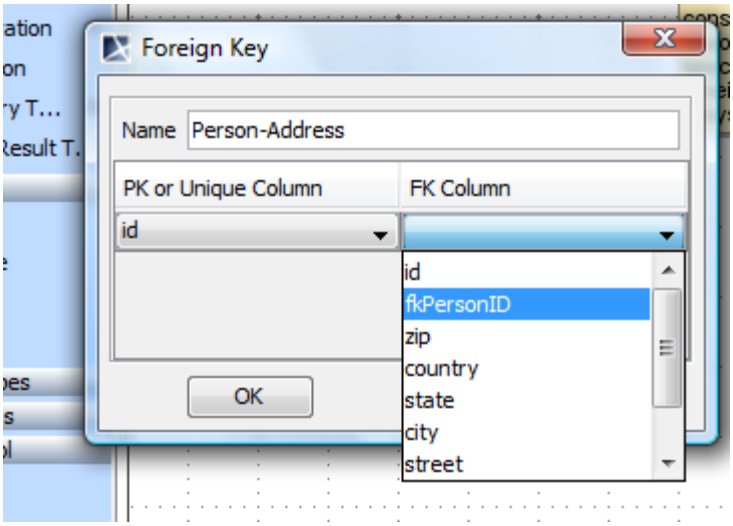
Foreign key example.

Foreign keys describe relationships between two tables. At the detailed understanding level, foreign key is a constraint on the (group of) columns in the source / referencing table, such that for each row in the source table their value combination (tuple) is equal to the value combination (tuple) of the (group of) columns for some row in the target / referenced table.

Foreign keys also have the two ways to be modeled. The main way is described below.

The main way to model foreign keys is to draw association relationship from the referencing table to the referenced table. The relationship can be simply draw in the diagram from the smart manipulator or from the button in the diagram toolbar.

When the FK association is drawn, the following dialog pops up.



Foreign key dialog.

Note that you have to have the necessary columns in the tables (PK or unique columns in target table, referencing FK columns in the source table) before drawing the FK relationship. In this dialog, select the referenced columns (of the target table) in the first column of the table, and corresponding referencing columns (of the source table). Additionally, foreign key name can be specified.

When dialog is OK'd, foreign key association is created; «FK» stereotype is applied on the referencing association end and the selected column information is stored in tags.

If foreign key information has to be changes, this is done in the Specification window of the FK property. Besides the standard SQL element properties foreign key has the following properties available in the Specification window.

Property name	Description
Inline	The same functionality as for the explicitly modeled PK, unique constraints.
Deferable	
Initially Deferred	
Enabled	
Match	Specifies how the columns in the referenced table and columns in the referencing table are matched (SIMPLE, FULL, PARTIAL match).
On Delete	Referential integrity enforcement action that is performed when the data in the referenced table is deleted (NO ACTION, RESTRICT, CASCADE, SET NULL, SET DEFAULT).
On Update	Referential integrity enforcement action that is performed when the data in the referenced table is updated (NO ACTION, RESTRICT, CASCADE, SET NULL, SET DEFAULT).
Referencing Members	Member columns of the constraint (from the same table where constraint is located). FK constrains values of these columns to point to the data in the referenced tables.
Referenced Members	The set of the columns in the referenced (target) table, to which referencing columns are matched. There are 6 ways to specify this set - choose one.
Referenced Table	Referenced Members field explicitly lists the target columns.
Referenced Unique Constraint	Referenced Unique Constraint / Index points to the constraint or index in the target table, and referenced member columns are members of this constraint / index.
Referenced (by Name) Unique Constraint	(by Name) option is used when constraint / index is no explicitly modeled with model element but is just a marking on the column.
Referenced Unique Index	Referenced Table always points to the target table of the FK (field is not editable, to change it, reconnect the association). If the referenced members column list is not specified in any other way, then referenced columns are taken from the PK of the referenced table.
Referenced (by Name) Unique Index	

The alternative way of modeling a foreign key is creating a UML constraint with the «ForeignKey» stereotype applied. This way is less desired than the main way, because it does not visualize relationship between tables. It is just a constraint in the table. This method may be used when human-readability is not critical, e.g., when database layout is generated with some custom automated script / transformation in the model.

To create a constraint with the «ForeignKey» stereotype

1. Select a table in the Containment tree.
2. Do one of the following.

- Right-click the selected element and from its shortcut menu select **New Element** > **Explicit Foreign Key**.
- Open the **Explicit Foreign Keys** tab in the table's Specification window.

Besides the standard SQL element properties and properties that are available for other explicit constraints (that is, PK, unique, check constraints), explicit foreign key has the following properties available in the Specification window.

Property name	Description
Match	Specifies how the columns in the referenced table and columns in the referencing table are matched (SIMPLE, FULL, PARTIAL match).
On Delete	Referential integrity enforcement action that is performed when the data in the referenced table is deleted (NO ACTION, RESTRICT, CASCADE, SET NULL, SET DEFAULT).
On Update	Referential integrity enforcement action that is performed when the data in the referenced table is updated (NO ACTION, RESTRICT, CASCADE, SET NULL, SET DEFAULT).
Referencing Members	Member columns of the constraint (from the same table where constraint is located). FK constrains values of these columns to point to the data in the referenced tables.
Referenced Members	The set of the columns in the referenced (target) table, to which referencing columns are matched. There are 6 ways to specify this set - choose one. Referenced Members field explicitly lists the target columns. Referenced Table field just specifies the target table, referenced columns are then taken from the PK of the table. Referenced Unique Constraint / Index points to the constraint or index in the target table, and referenced member columns are members of this constraint / index. (by Name) option is used when constraint / index is no explicitly modeled with model element but is just a marking on the column.
Referenced Table	
Referenced Unique Constraint	
Referenced (by Name) Unique Constraint	
Referenced Unique Index	
Referenced (by Name) Unique Index	

Nullability constraint

Note
SQL NOT NULL constraint (if modeled explicitly, which is rare) is modeled as UML Constraint with «NotNullConstraint» stereotype applied.

Nullability, or NOT NULL constraint forces the condition that the column must have value. Implicit NOT NULL constraint is modeled with the **nullable** field of the column (set nullable=false to specify NOT NULL). This is a usual and quick way to model these constraints.

Usually, there is no need to model these constraints explicitly - create a separate model element for them. But in the more complex cases these constraints can be created by hand and the «NotNullConstraint» stereotype applied on them. This allows specifying non-inline constraints, or named constraints, or deferred constraints or inactive constraints.

NOT NULL constraint does not have any additional properties in the Specification window besides the properties that all table constraints have.

Assertion

Note
SQL Assertion is modeled as UML Constraint with «Assertion» stereotype applied.

Assertion constraints are very similar to check constraints, but instead of being local to the table, they are global to the database. Assertions check some condition that must hold through several tables. Assertions are modeler as explicit constraints; there is no shorthand modeling form - assertion is always an explicit UML constraint.

To create an assertion

1. Select a schema or a database element in the Containment tree.
2. Right-click the selected element and from its shortcut menu select **New Element** > **Assertion**.

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

Property name	Description
Search Condition	The assertion body condition
Constrained Tables	List of the tables on which assertion runs

Triggers

> **Note**
> SQL Trigger is modeled as UMLOpaqueBehavior with the «Trigger» stereotype applied.

Trigger describes some action that must be performed when some particular data manipulation action is being performed in the database table. Trigger can be fired when data is added, deleted or changed in the table and perform some action (update some other table, calculate additional values, validate data being updated or even change the data that is being updated).

Trigger is always defined for some table. You can define triggers in the **Triggers** tab of the table Specification window. Trigger has an event type (on what actions trigger is fired, that is, on insert, on update, or on delete), action time (before, after, instead of), and an actual body describing the actions.

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

Property name	Description
Action Time	Specifies moment of time when trigger action is performed (before the specified event, after event, insteadof event).
On Insert	The event that causes trigger firing.
On Update	
On Delete	
Trigger Column	List of columns, which causes trigger fire on update (must be from the same table as trigger is located). Used with On Update triggers to specify that only some column updates cause trigger fire.
Language	Trigger implementation language (should be SQL).
Body	Trigger body text (operations that are performed on trigger fire).
Time Stamp	Trigger creation timestamp.
Action Granularity	Specifies whether trigger fires once per executed statement, or once per each affected row.
When	Specifies additional precondition for trigger firing.
New Row	These fields can be used for defining variable names for holding new row / table values and old row / table values - for referencing in trigger body. REFERENCING ((NEW|OLD) (TABLE|ROW) AS <name>)+
New Table	
Old Row	
Old Table	
Additional Actions	Additional action statements. This option is rarely used - it is non-standard and supported only by some databases. Usually, triggers have just one body.