| BR-001  Keys | |
| --- | --- |
| **Business Requirement (BR)** | **Functionality** |
| [BR-001-1](#_BR-001-1:_Table_without) | Table without primary key |
| [BR-001-2](#_BR-001-2:_Primary_key) | Primary key with one column (Euss assigned id) |
| [BR-001-3](#_BR-001-4:_Primary_key) | Primary key with one column (User assigned id) |
| [BR-001-4](#_BR-001-4:_Primary_key_1) | Primary key with one column (Native id) |
| [BR-001-5](#_BR-001-5:_Primary_key) | Primary key with multiple columns |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Description

An entity can embed a primary key or not. This primary key is made of one more fields.

In addition each field can be:

* Automatically assigned by EUSS
* Assigned by the user
* Native from the database

# Technical specifications

## Primaries keys

An **Attribute** is an element of the mapping representing how a property of a class is mapped to a field of a table. Each of them can be a part of the primary key.

This means that the **Attribute** element contains 2 optional values:

* isPrimaryKey indicates if the attribute is a part of the primary key
* generator indicates how the id value is generated

<Attribute name="Lastname" field="Lastname" db-type="AnsiString"

isPrimaryKey="true" generator="assigned" />

<Attribute name="Firstname" field="Firstname" db-type="AnsiString"

isPrimaryKey="true" generator="native" />

If the entity doesn’t contain any field as primary key, EUSS can only execute **select** and **insert** queries. The **delete** and **update** queries are disabled.

If an attribute is a part of a primary key, the **not-null** constraint is automatically set to true.

## Foreign keys

An entity can reference another entity using a relationship. This relationship is based on foreign keys.

# Business requirements

## BR-001-1: Table without primary key

This BR is designed to check that EUSS is able to load or serialize data into a table which doesn’t have any primary key.

The delete and update operations are disallowed and must thrown an exception.

We have to insert some data in the table using EUSS and check that the values are correctly inserted using both EUSS and ADO .NET.

The following operations must succeed:

* Serialize
* Load

The following operations must fail:

* Delete
* Update

## BR-001-2: Primary key with one column (EUSS assigned id)

This BR is designed to check that EUSS automatically generates a GUID value for the id. Then we have to check that the value of the id is really a GUID.

Moreover, the value of the id before serialization must be the same after serialization.

The following operations must succeed:

* Serialize
* Load
* Delete
* Update

## BR-001-3: Primary key with one column (User assigned id)

This BR is designed to check that the user can set the value of the primary key. Then we have to define an attribute as primary key.

We have to get the value of the field after serialization without using EUSS and check that this value is the expected value.

If value isn’t defined (e.g.: string.Empty, double.NaN …) the serialization operation must fail.

The following operations must succeed:

* Serialize
* Load
* Delete
* Update

## BR-001-4: Primary key with one column (Native id)

This BR is designed to check that EUSS get the value affected by the database.

We have to check that the value of the property is correctly set after serialization. In addition, we check that the value is correct without using EUSS (e.g.: ADO .NET)

The following operations must succeed:

* Serialize
* Load
* Delete
* Update

## BR-001-5: Primary key with multiple columns

This BR is designed to check that EUSS is able to store and load data if the primary key is made of more than 1 column.

All columns don’t have the same type of id.

The following operations must succeed:

* Serialize
* Load
* Delete
* Update