Technical Note

Normalizing repository tables with the Normalization wizard

Author: R&D Department

Publication date: December 8, 2008

Revision date: December 2010

Normalizing repository tables with the Normalization wizard

Overview

One of the main advantages of REP++ is the centralization of the metadata definitions in the REP++ repository. If an organization uses a database column called SIN (social insurance number) in the REP++ repository, then this same, unique definition will be reused everywhere the database column is used. Another main advantage of REP++ is the way it completely automates the management of complex father-child relationships.

In order to achieve the centralization of the metadata in the repository and assume the proper management of relationships, REP++ requires a normalized set of tables. Unfortunately, existing databases are seldom ideal or normalized.

In order to gap the bridge between a non-normalized database and the requirements of the repository, REP++ integrated a new tool, the **Normalization Wizard**, whose goal is to help you sort out the duplications or incoherences that often arise in existing databases so that they are taken into account into REP++.

This document describes the specific problems that are addressed by the normalization wizard and how it can resolve them simply and intuitively.

Normalization issues

REP++ requires a certain level of normalization from its set of tables in order to manage efficiently the metadata and their relationship. In particular, REP++ does not allow for columns with same name but different types: a column name must represent one and only one entity, and all columns sharing the same name in the database should have the same definition.

The initial metadata of a column is often imported from an existing source database using the REP++ Update wizard. If the source database is normalized, the import creates all the equivalent tables and columns in the repository. Unfortunately, source databases, built and modified over time, do not always attain the level of normalization required by REP++. For instance, they may contain duplications or inconsistencies that make them vulnerable to problems such as data integrity. To try to correct the database itself would generally prove very difficult to achieve, if even possible.

This is where the REP++ normalization wizard steps in: it identifies the columns that are duplicated or inconsistent, and helps you map them to appropriate columns or to new columns of the repository. You reach the appropriate level of normalization for REP++ without any changes to the database itself; REP++ manages the rest with no further intervention.

Problems addressed by the normalization wizard

There are two types of issues that the normalization wizard addresses:

A. **Columns with same name but different definitions**. The existing database contains several columns with the same name but the column definitions are different.

B. **Columns with differing names that represent the same entity**. The existing database contains columns with different names that represent the same thing. This leads to 2 problems:
   - Primary key and foreign key columns can have different names in database tables;
   - Useless duplication of information.

Case A: Columns with same name but different definitions

The example below describes the case where a column name has more than one definition in different tables.

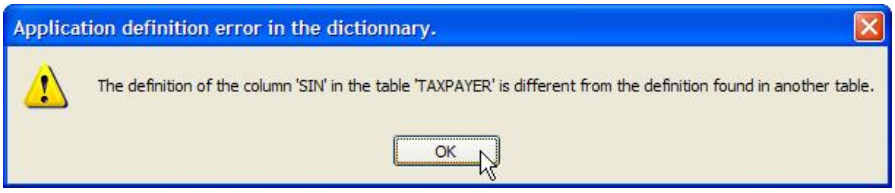
Employee

Column Name	Condensed Type	Nullable
EmployeeID	bigint	No
FirstName	varchar(50)	No
LastName	varchar(50)	No
SIN	char(11)	No

TaxPayer

Column Name	Condensed Type	Nullable
TaxPayerID	bigint	No
FirstName	varchar(50)	No
LastName	varchar(50)	No
SIN	int	No

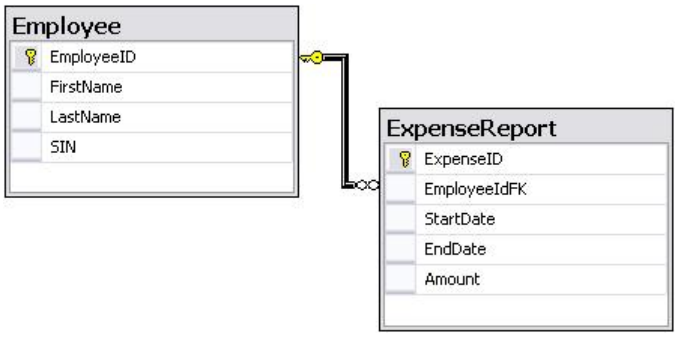
If you try to import the definitions of the tables *Employee* and *TaxPayer* using the REP++ Update wizard, you will get the following error message.



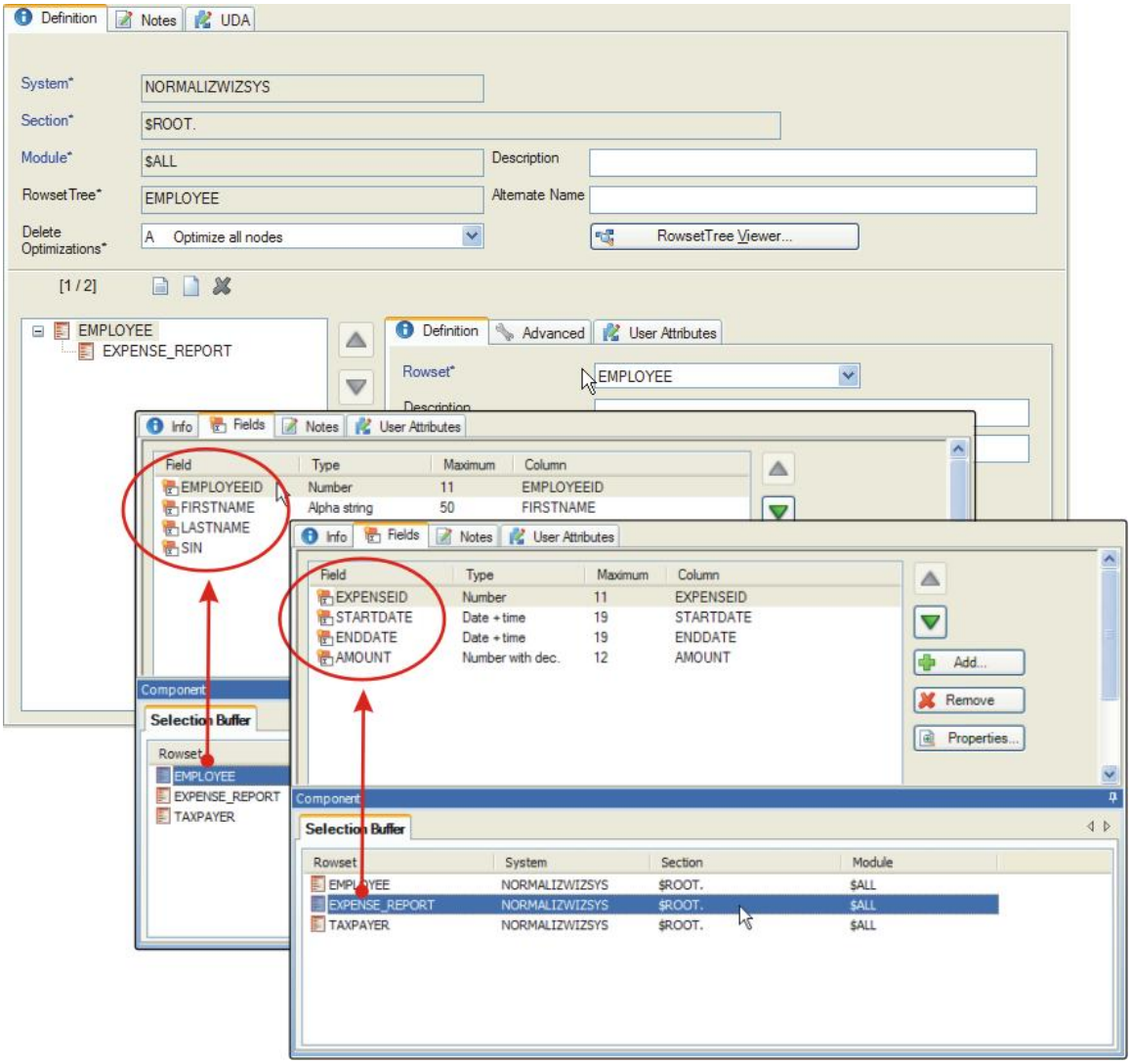
In this example, the SIN column actually represents the same entity but they are defined differently, one as a string type, the other as an integer type. REP++ requires that the name of a column in the database be unique and represent a single entity in the repository.

Case B: Columns with differing names for primary and foreign keys

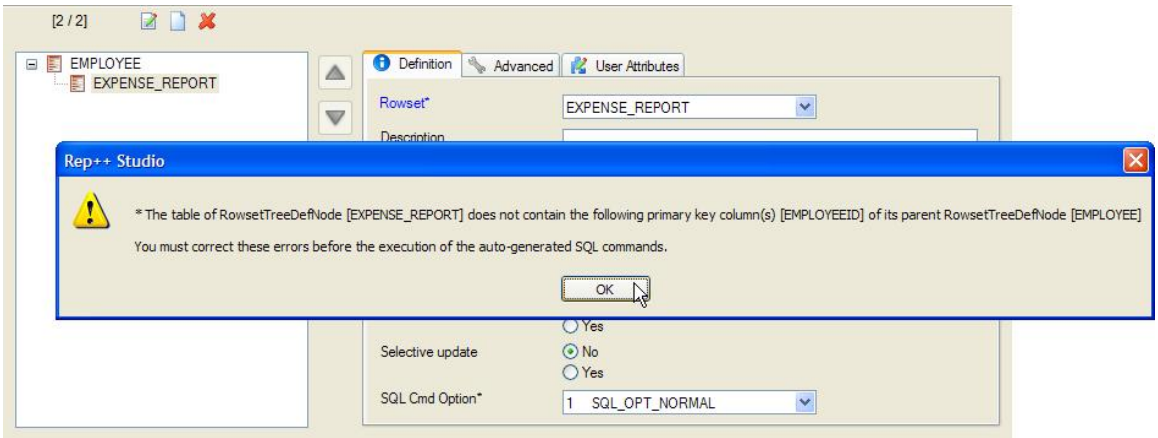
The main problem arises when the database schema does not respect the naming rule that primary keys and foreign keys should have the same name. For instance, in the following tables, the primary key of the **Employee** table (EmployeeID) is different than the foreign key in the **ExpenseReport** table (EmployeeIdFK) even though they represent the same thing.



The definition of a REP++ RowsetTreeDef that represents the employees and their expense report is shown below.



When you attempt to save this REP++ RowsetTreeDef, you will get the following warning complaining that the EmployeeID column could not be found in the ExpenseReport table.



You need to resolve this mismatch in order for REP++ to manage the relation correctly.

As for the useless duplication of metadata, sometimes columns with different names in different tables really could be described by the same entity in the repository. For instance, columns Description, Desc, Memo, Notes that share the same definition throughout a database could more simply be described by a single entity in the repository. This simplification would decrease the number of useless definitions, thereby reducing the number of entities required to describe a system.

Correcting the problems

There are a limited number of approaches to resolve those issues. One is to modify the database itself, which can be, for all practical purposes, at least very complex if not impossible. The other is to make the REP++ repository aware of those issues, so that they can be taken into account automatically. This can be done manually or using the normalization wizard.

Modifying the source database

In the case of the multiple definitions problem, the solution would be to modify the database so that all columns with the same name must have the same definition, or to change the names of unrelated columns that have the same name. In the case of differing names for primary and foreign keys, you would need to modify the schema so that a primary key column and all its referencing columns have the same name.

Modifying the database is however quite complex and risky to implement since it is sometimes difficult to assess the impacts of a change on an already existing database, especially when it involves several dependent systems.

Mapping columns manually

Another solution would be to make the changes in the REP++ repository itself and let REP++ manage them. With REP++, you indirectly deal with this kind of situation by mapping a logical column in the repository to a column's physical name that is used when generating SQL commands that interact with the database management system. You map a column name in the repository to a column name in the database in the tables editor.

Case of same name but multiple definitions

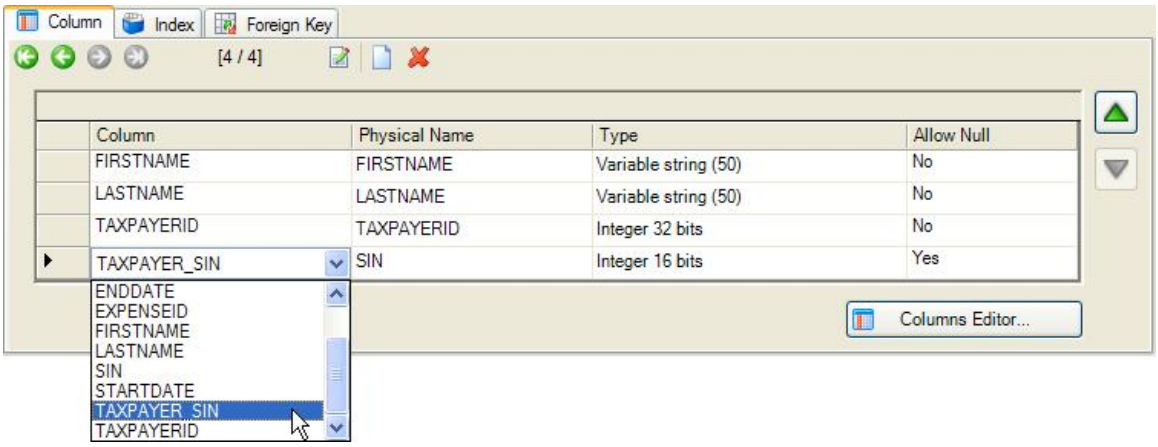
In the case of multiple definitions problem, you need to create a column with a unique name to use with one of the column definitions.

MAPPING A UNIQUE COLUMN NAME TO A DATABASE COLUMN

You need to choose which one of the column definitions you want to "rename" in the repository.

To map a unique repository column name to a database column:

1. Create a new column in the repository with a type that corresponds to the type of the chosen column causing the problem in the database.
2. In the tables editor, select the table that will contain the new column.
3. Click the **Column** tab to display the list of columns for the table.
4. Locate the chosen database column under **Physical Name**, then under **Column**, select your new column from the list.



In the example for the SIN column described before, you would create a new column with a unique name, for instance a TAX_PAYER_SIN column of type INT for the TaxPayer table. You would then associate this repository column (logical name) to the SIN database column (physical name) in the TaxPayer table. When REP++ sees the SIN database column, it will use instead the TAX_PAYER_SIN repository column, and vice-versa.

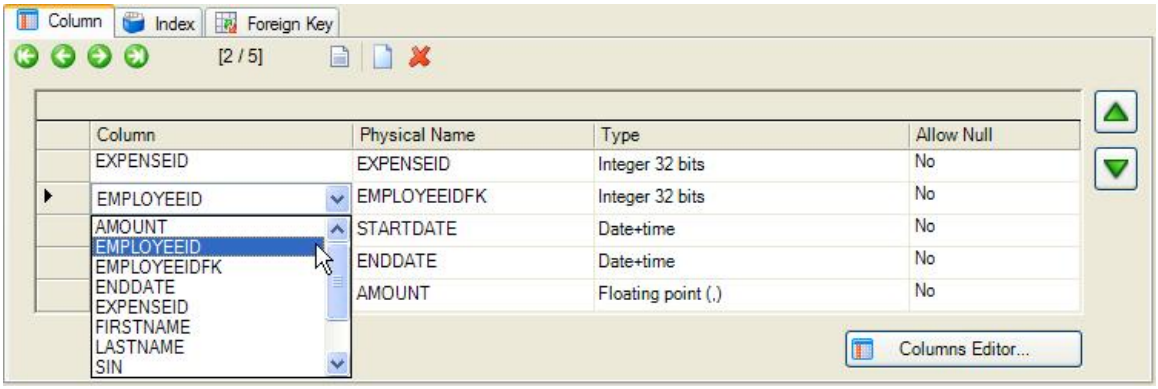
Case of columns with differing names for primary and foreign keys

In the case of columns that differ in name for primary and foreign keys, you can assign the name of the primary key column to the logical name of the foreign key column in the descendant tables.

Mapping a foreign key column to its father primary key column

To map a foreign key to its father primary key column:

1. In the tables editor, select the child table that contains the foreign key to change.
2. Click the **Column** tab to display the list of columns for the table.
3. Locate the foreign key of the table under **Physical Name**, then select the name of the primary key of its father table from the list under **Column**.



In the database, the foreign key column name will remain the same, but when REP++ sees the foreign key, it will use instead the logical name corresponding to the primary key of its father table.

Using the Normalization wizard

The manual method for mapping new or existing repository columns to database columns is a simple way to correct the source database's duplications and inconsistencies. Unfortunately, it does not give you a global view of the modifications you need to do. In both cases, you can achieve the same result using the **Normalization wizard**, which provides a user-friendly interface that will display a global view of what needs to be done and help you perform the necessary mappings.

The normalization wizard

The normalization wizard was created to facilitate the correction of issues such as multiple definitions and differing names for primary/foreign keys. By selecting a table or set of tables to analyze, the wizard presents, in an intuitive user interface, a comparative view of the repository and the database tables and columns. It displays, in one, easy-to-understand window, the columns attached to the tables as stored in the repository and the columns attached to the tables in the database. The wizard highlights inconsistencies so that you can right away make the necessary modifications.

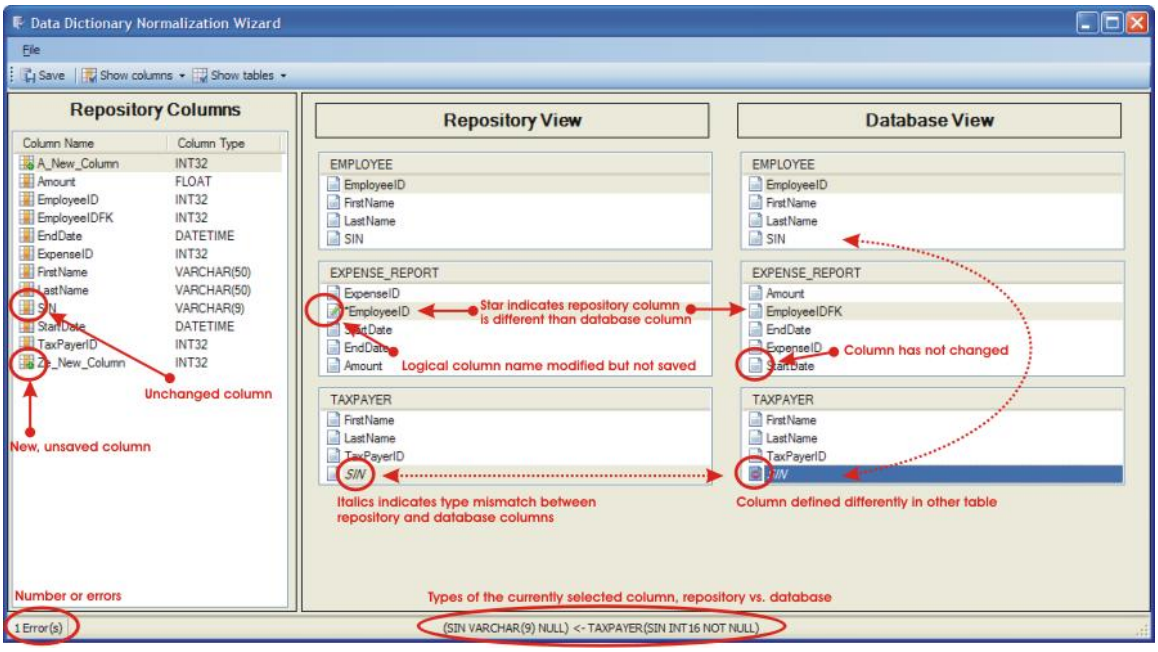
When to use the normalization wizard

You use the Normalization wizard when you start a new information system using REP++ from an existing database. You would first run the Update wizard to import the tables of the database into the repository. If the Update wizard reports errors such as duplicate columns, then you use the Normalization wizard to help you make the necessary corrections.

Remember that the normalization wizard does not change in any way your database schema. Instead, it helps you create and/or map columns of the repository (the logical columns) to existing physical columns in the database, thereby achieving the required level of normalization for REP++.

The normalization wizard environment

The normalization wizard environment is shown below.



Repository Columns

On the left, the **Repository Columns** displays the list of columns that are associated with one of the analyzed tables, along with their type. The following icons describe the state of the column.

The column has been created in memory but has not been saved.

The column has not changed since it was loaded.

To show all columns, click **Show columns** on the toolbar, then **Show all columns**.

Repository View and Database View

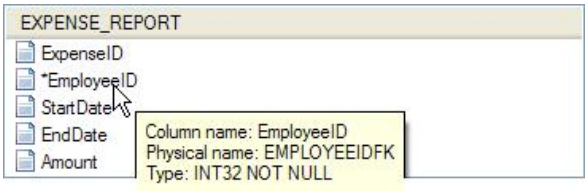
On the right, the **Repository View** and the **Database View** show a comparative view of the repository tables and their database counterparts. By default, this view includes all the analyzed tables. To display only tables in error, click **Show tables** on the toolbar, then **Only tables with errors**.

Note

If a table exists either in the repository or the database but not in both, it will only be displayed in the view corresponding to its location.

Table controls

The repository and database views use a number of table controls to represent their tables.A table control displays a repository or database table's columns. Moving the mouse over a column displays a tooltip providing additional information regarding the column's name, physical name and type. Selecting a repository column selects its corresponding database column (and vice-versa), and the types of both versions are displayed in the main status bar.



The state of a column is expressed as follows:

- If there is a difference between the type of the database and repository versions of the column, the column name turns into italics.
- If the column's logical and physical names are different, it is prefixed with an asterisk (*) in the repository view.

The following icons are used to describe the state of a column.

The table column has not changed since it was loaded.

The table column has changed in memory but has not been saved.

The database table column has the same name as another column but a different definition.

The figure below shows you an example of how the presence of a column name with different definitions would appear in the wizard. The status bar indicates one error. It also shows that the SIN column of the TaxPayer table in the repository has a type that is different from its counterpart in the database.

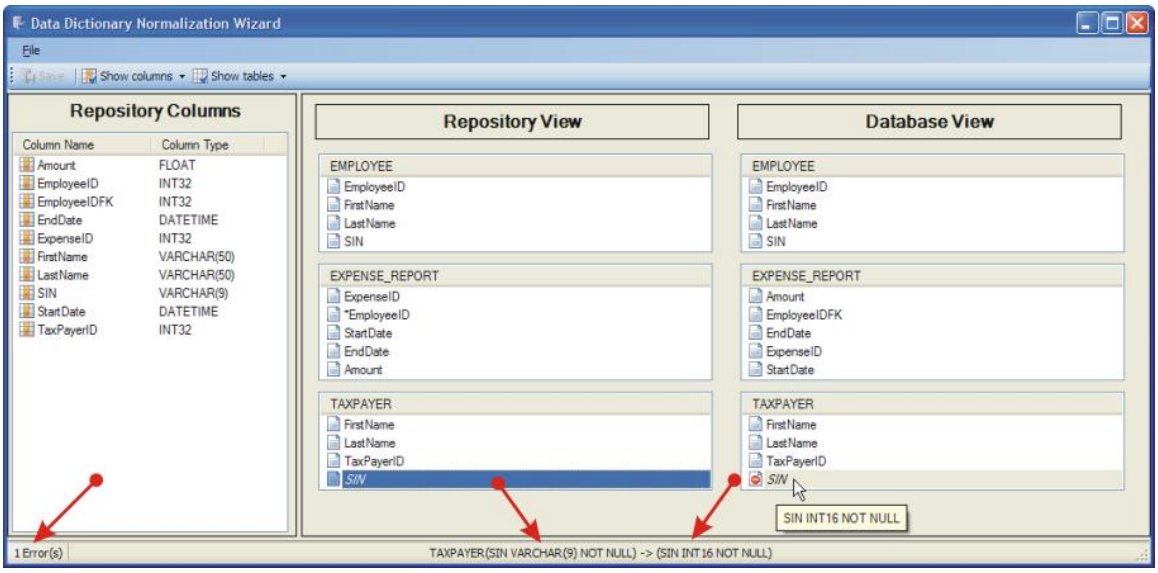
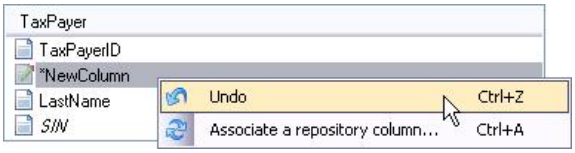


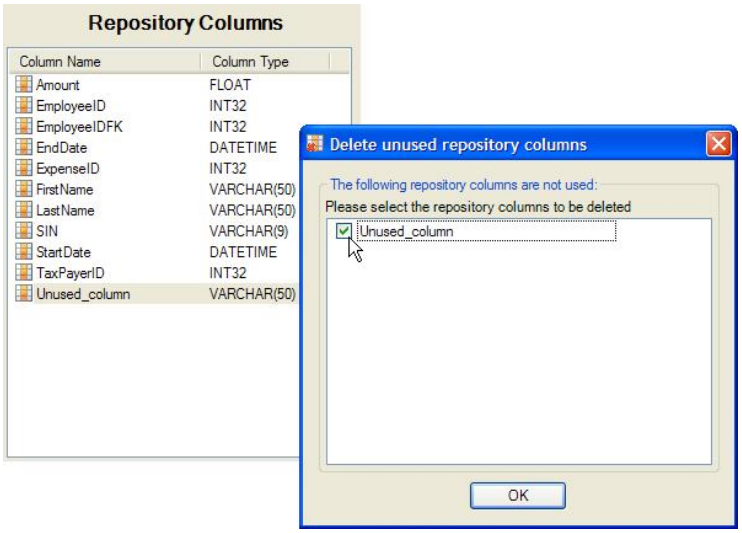
Table controls handle drag-and-drop. You can rename a column by dragging it from one table control to another or from the **Repository Columns**. Table controls also provide a context sensitive menu that enables you to:

- Rename a column.
- Undo a modification (if applicable).



Unused columns window

When you save your changes, the normalization wizard checks if there are any repository columns that are no longer in use. If there are unused repository columns, the wizard opens the *Delete unused replository columns* dialog box where you can select the columns to delete.



Important

All changes made by the normalization wizard are applied to the metadata of the repository only. **Your database schema is not modified in anyway**.

Using the normalization wizard

Starting the normalization wizard

You normally run the Normalization wizard after you use the Update wizard to update the repository tables.

STARTING THE NORMALIZATION WIZARD

To start the Normalization wizard:

1. On the **Tools** menu of the REP++ repository editor, click **Wizards**, then **Normalization wizard**. The *Select tables* dialog box opens.

2. Select the tables to be analyzed and displayed. Click **OK**.

The *Data Dictionary Normalization Wizard* window will open.

Solving the case of multiple definitions

When there are multiple definitions for a database column name, you need to associate a type-compatible column of the repository to the database column. To do so, you can use an already existing column or create a new column.

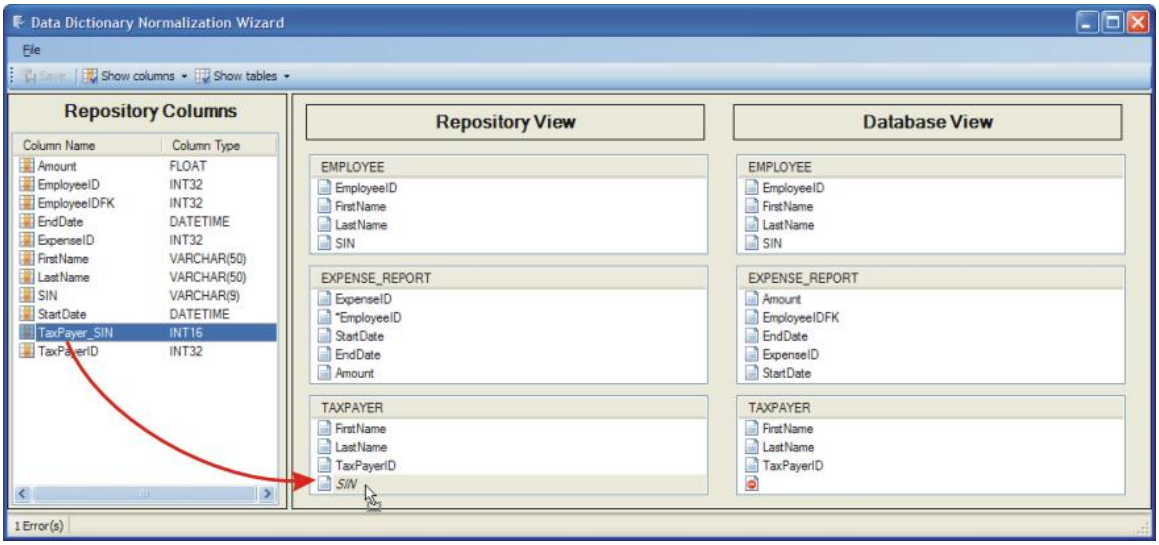
When a type-compatible column exists

USING AN EXISTING TYPE-COMPATIBLE REPOSITORY COLUMN TO MAP TO A DATABASE COLUMN DEFINED MORE THAN ONCE

You can use the drag-and-drop or the context menu.

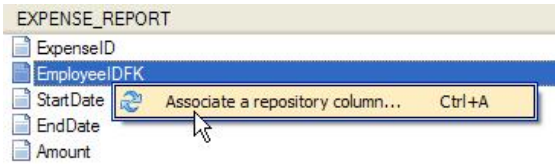
To map a repository column using drag-and-drop:

- Drag the existing type-compatible column over the column in the Repository View. You can drag a column either from the list of columns or from a table definition.



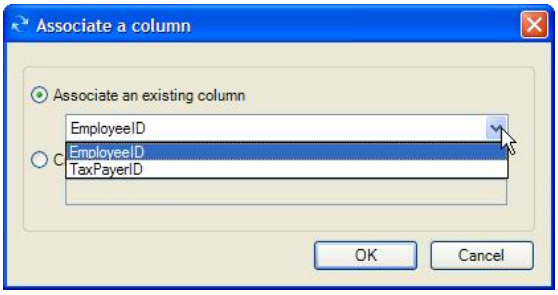
To map a repository column using the context sensitive menu:

1. Right-click on the repository column and click **Associate a repository column**.



The *Associate a column* dialog box opens.

2. Click **Associate an existing column** and choose one of the proposed columns. Click **OK**.



Note

If there are no type-compatible columns within your system, this option will be disabled. See section When no type-compatible column exists.

3. Save your modifications.

The chosen column name will now appear in the Repository View.

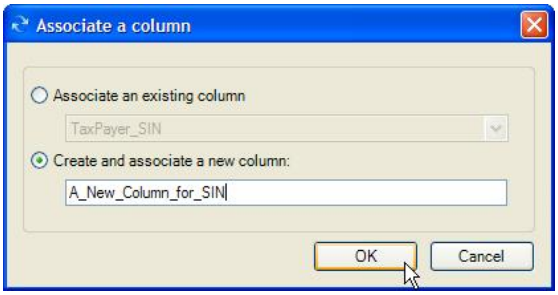
When no type-compatible column exists

If no column has a compatible type, the wizard provides a context-sensitive menu that allows you to easily create a compatible column by simply providing the column name.

Creating a new type-compatible column to map to a database column defined more than once

To create a new column in the normalization wizard:

1. Right-click on the column in error and select **Associate a repository column**. The *Associate a column* dialog box opens.
2. Click **Create and associate a new column** and enter the name of the new column. Click **OK**.



Note

If the column name is invalid or the column already exists, the **OK** button is disabled.

3. Save your modifications.

The chosen column name will now appear in the Repository View.

Solving the case of columns with differing names for primary and foreign keys

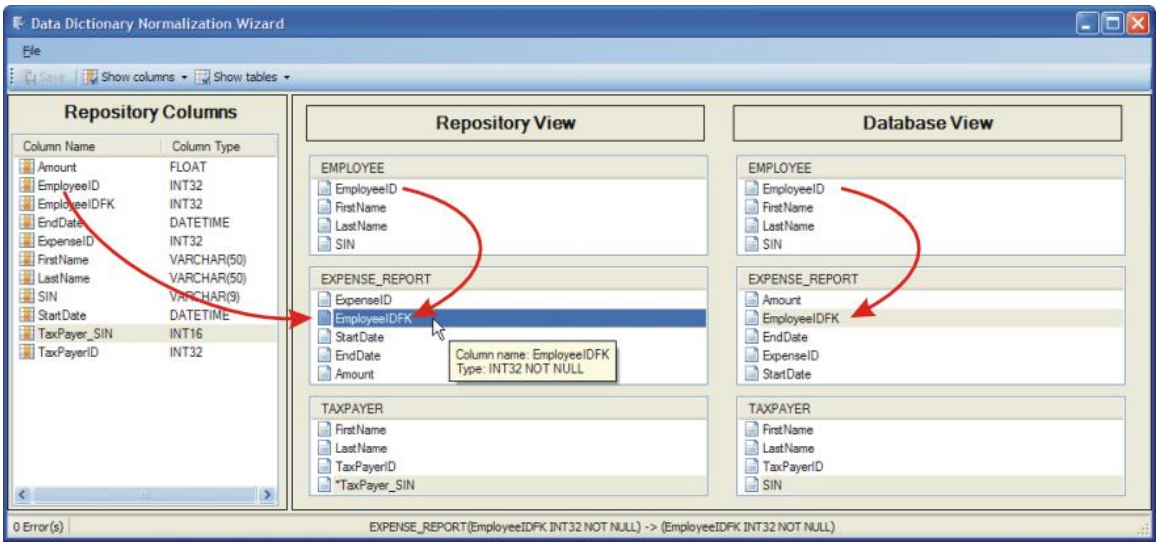
Fixing the different primary key and foreign key column names using the normalization wizard only involves selecting the appropriate column.

ASSOCIATING A FOREIGN KEY COLUMN TO ITS FATHER KEY IN THE REPOSITORY

You can use the drag-and-drop or the context menu.

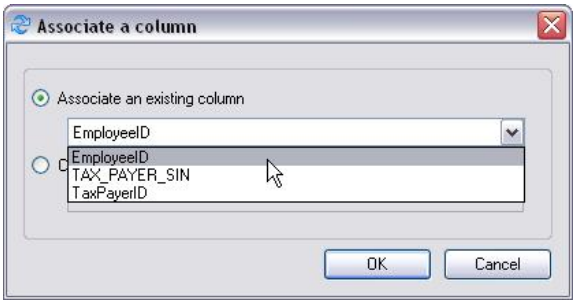
To map a repository column using drag-and-drop:

Simply drag the primary key column either from the **Repository Columns** or from the parent table, and drop it on the foreign key column in the child table.



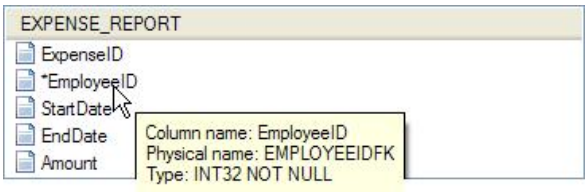
To map a repository column using the context sensitive menu:

1. Right-click on the foreign key column and select **Associate a repository column**. The *Associate a column* dialog box opens.
2. Click **Associate an existing column** and select the primary key column of the parent table. Click **OK**.



3. Save your modifications.

The selected column name appears in the Repository View of the wizard. The tooltip indicates that the EmployeeID column is now mapped to the EmployeeIdFK.



Once you are done with the wizard, REP++ will take charge of using the appropriate database tables mapped with the repository tables. No more intervention on the developer's part is required unless the source database is modified (column modified or added), in which case you would run the update wizard followed by the normalization wizard to detect if new mappings are necessary.