

Persistence Designer

User's Guide

Version 10g Release 3 (10.3)

PERSISTENCE DESIGNER.....	3
ADDING PERSISTENCE SUPPORT	4
PERSIST AS COLUMNS OF A TABLE.....	5
PERSIST ENTIRE MESSAGE AS XML	6
DATABASE TABLE DESIGN.....	7
SPECIFYING FIELD PROPERTIES.....	8
SPECIFYING SECTION PROPERTIES	9
ADDING FIELD TO PERSISTENCE DESIGNER.....	9
ADDING SECTION TO PERSISTENCE DESIGNER	10
REMOVING FIELD/SECTION FROM PERSISTENCE DESIGNER.....	10
IMPORTING FIELD STRUCTURE FROM EXTERNAL SOURCES	11
<i>Import Field Structure From Database Tables</i>	<i>11</i>
<i>Import Field Structure from an XML Schema File.....</i>	<i>13</i>
<i>Import Field Structure from a DTD File</i>	<i>15</i>
QUERIES.....	16
DEFINING QUERIES IN PERSISTENCE DESIGNER.....	16
REMOVING QUERIES	18
INVOKING QUERIES FROM SIMULATOR.....	18
DATABASE MAPPING.....	19
UPDATING PERSISTENCE DESIGNER FROM INTERNAL MESSAGE.....	20
REINITIALIZING PERSISTENCE DESIGNER.....	20
UPDATING PERSISTENCE DESIGNER.....	21
REMOVING PERSISTENCE DESIGNER.....	21
CONFIGURING THE DATA SOURCE.....	21
POPULATING TABLES	24
RUNTIME CONFIGURATION	25
OR MAPPING RULES AND BEHAVIOR.....	26
General Restrictions.....	26
Auto Generated Keys.....	26
Foreign Keys.....	26

Persistence Designer

Persistence Designer can be used to persist an internal message to a database. Queries can also be defined in the persistence designer which can be used to retrieve data that has been persisted in the database. Mapping between fields of the internal message and the fields in the tables can be specified using the persistence designer.

See Also:

[Adding Persistence Support](#)

[Database Table Design](#)

[Queries](#)

[Database Mapping](#)

[Updating Persistence Designer From Internal Message](#)

[Removing Persistence Designer](#)

[Configuring the Data source](#)

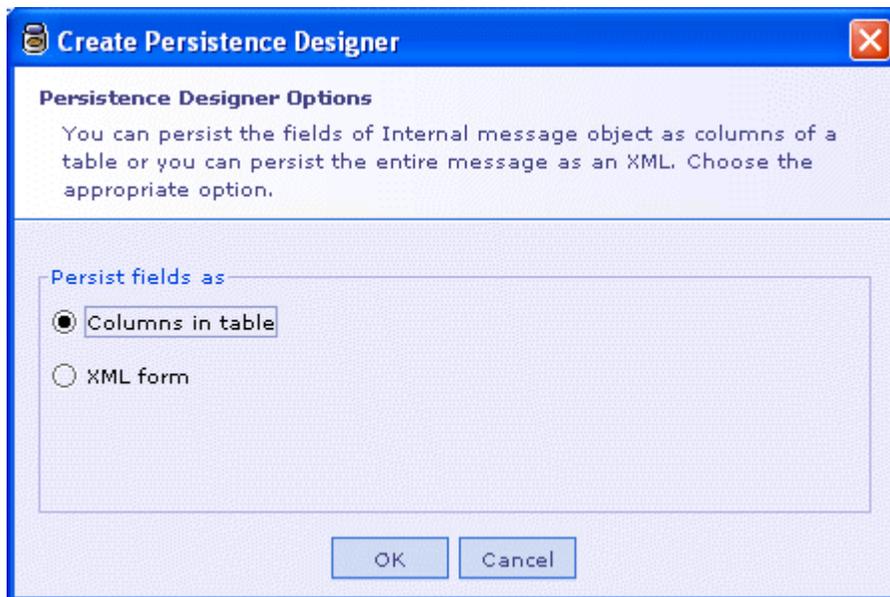
[Populating tables](#)

[Runtime configuration](#)

[OR Mapping Rules and Behavior](#)

Adding Persistence Support

1. Right click the Internal Message and select **Add Persistence Designer** menu item from the context menu.
2. 'Create Persistence Designer' dialog will be displayed as shown below.



Select the manner in which you want to persist the fields of the internal message. You can

- persist the fields as columns of a table (or)
- persist the entire message as an XML.

See Also:

[Persist as Columns of a table](#)

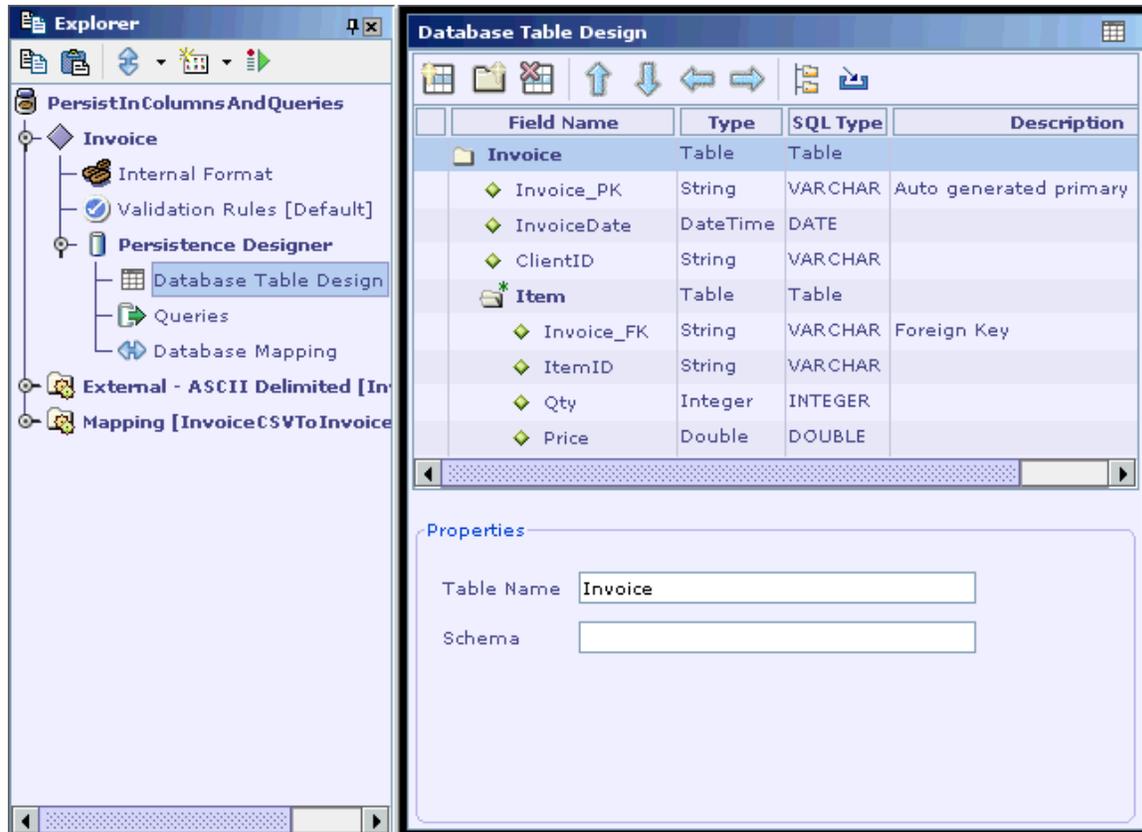
[Persist Entire Message as XML](#)

[Updating Persistence Designer From Internal Message](#)

[Removing Persistence Designer](#)

Persist as Columns of a Table

If this option is chosen the persistence designer added will appear as shown below:



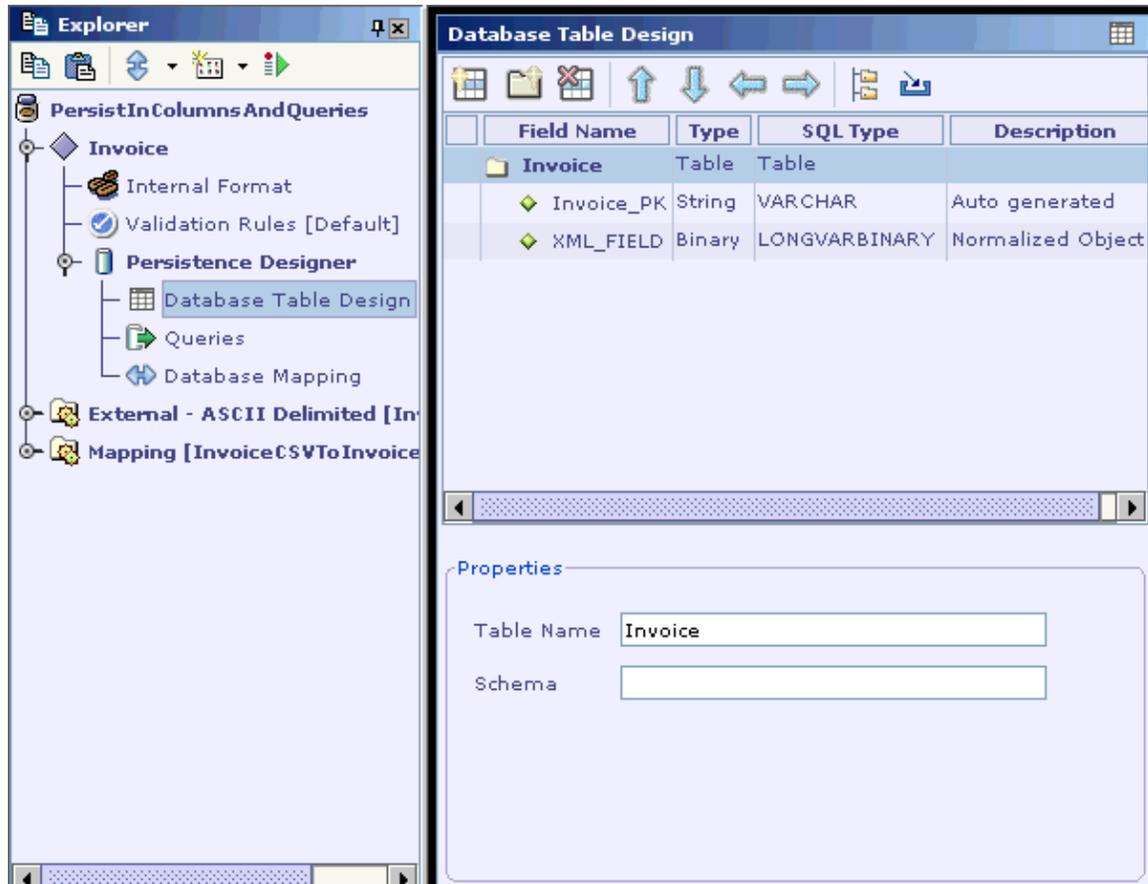
In this mode you can map the fields of the NO to columns of one or more tables.

See Also:

[Persist Entire Message as XML](#)

Persist Entire Message as XML

If you choose the second option Persistence designer will be added as shown below



A table structure with just the primary key for the NO and 'XML_FIELD' column will be created. The NO is persisted as an XML blob in XML_FIELD column.

Though you want the NO to be stored as XML, you may still want to query based on some other important fields. To do this, you can add some more columns to the database table and map them to fields of the NO. The modality for this is same as in the normal persistence case (option 1).

The name of 'XML_FIELD' column should not be changed.

While loading NO from database (queries) the user defined columns are ignored and the NO is deserialized from the XML. During creation (insert) the user defined columns are populated as per the mapping specified.

Note:

The following are automatically done when you add a Persistence Designer.

The table structure (fields) and the master-detail relationship between them are automatically created. It assumes that the primary keys and the foreign keys have to be synthesized (not present in NO). In many cases this should be sufficient, particularly if you are interested in just persisting and retrieving normalized objects. In case you already have the schema, you can treat this as starting point, or you can completely delete all the DB fields and restart from scratch.

Database mapping from NO to DB is automatically synthesized. You can modify the mapping if you want to.

See Also:

[Persist as Columns of a table](#)

Database Table Design

After the persistence designer has been created, properties for the fields (columns in a table) in the persistence designer can be specified in the Database Table Design UI. New fields can also be added to the Persistence Designer. Fields can also be imported from external sources.

See Also:

[Adding Field To Persistence Designer](#)

[Adding Section To Persistence Designer](#)

[Removing Field/Section From Persistence Designer](#)

[Specifying Field Properties](#)

[Specifying Section Properties](#)

[Importing Field Structure from External Sources](#)

Specifying Field Properties

Select a field. In the field properties panel you can set various properties.



The screenshot shows a 'Properties' panel for a field. It is divided into three columns. The first column contains a 'Length' text box with the value '36'. The second column contains three checked checkboxes: 'Not Null', 'Primary Key', and 'Auto Generate', with a text box containing 'uuid()' below them. The third column contains three checked checkboxes: 'Foreign Key', 'Foreign Key Table', and 'Foreign Key Column', with text boxes containing 'tbl_R_MsgExecu' and 'MessageExecuti' respectively.

Length	36	Not Null	<input checked="" type="checkbox"/>	Foreign Key	<input checked="" type="checkbox"/>
		Primary Key	<input checked="" type="checkbox"/>	Foreign Key Table	tbl_R_MsgExecu
		Auto Generate	<input checked="" type="checkbox"/>	Foreign Key Column	MessageExecuti
			uuid()		

The length of the field can be specified in the 'Length' text box.

You can specify if the field is a mandatory field by checking the 'Not Null' check box.

You can specify whether the field is the primary key for the table. If it is the primary key you can specify whether the key should be auto generated by designer.

For auto-generated fields you can associate a formula that will be executed to get the auto generated value. For instance, a formula like `uuid()` would generate unique key. You can also use this method to call custom functions that for instance, fetches unique key from an Oracle sequence.

The formula text box is visible when the 'Auto Generate' check box is enabled.

You can also specify if the field is a foreign key. In such a case you should specify the foreign key table and foreign key column.

The foreign key table and foreign key column are visible when the 'Foreign Key' check box is enabled. By default the 'Foreign Key' check box is disabled.

See Also:

[Specifying Section Properties](#)
[Database Table Design](#)

Specifying Section Properties

Each section present in the persistence designer corresponds to a table. Select a section. In the section properties panel you can specify the properties for the section.



The screenshot shows a 'Properties' panel with two input fields. The first field is labeled 'Table Name' and contains the text 'Invoice'. The second field is labeled 'Schema' and is currently empty.

You can specify the table name to be used. By default the section name is set as the table name.

You can specify the schema for the table. This is an optional property.

See Also:

[Adding Field To Persistence Designer](#)

[Specifying Field Properties](#)

[Database Table Design](#)

Adding Field To Persistence Designer

New fields can be added to the Persistence Designer in addition to the fields that are created from the internal message. Click the  button. A new field will be added. A field corresponds to a column in a table. You can specify properties for the field.

See Also:

[Specifying Field Properties](#)

[Adding Section To Persistence Designer](#)

[Removing Field/Section From Persistence Designer](#)

Adding Section To Persistence Designer

New sections can be added to the Persistence Designer in addition to the sections that are created from the internal message. Click the  button. A new section will be added. A section corresponds to a table.

See Also:

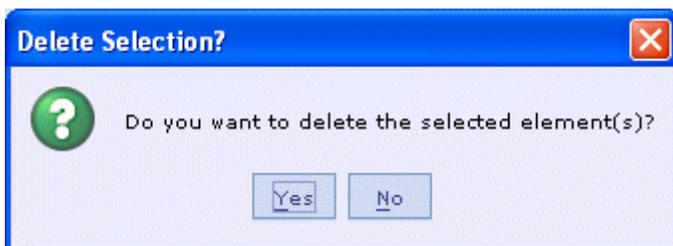
[Specifying Section Properties](#)

[Adding Field To Persistence Designer](#)

[Removing Field/Section From Persistence Designer](#)

Removing Field/Section From Persistence Designer

Select the field(s)/section(s) that is to be removed. Click the  button in the toolbar. The following dialog is displayed.



Click 'Yes' to delete the selected elements.

You can also select the field(s)/section(s) to be removed and press the 'Delete' key. The Delete Selection dialog will be displayed. Click 'Yes' to delete the selected elements.

Note:

The 'XML_FIELD' column present if 'Persist as XML' option has been chosen cannot be removed.

See Also:

[Adding Field To Persistence Designer](#)

[Adding Section To Persistence Designer](#)

Importing Field Structure from External Sources

The **Import Field Structure** icon  available in the Persistence Designer UI helps in defining the Persistence Designer by importing field structure from database tables, XML schema files and DTD files.

See Also:

[Import Field Structure From Database Tables](#)

[Import Field Structure from an XML Schema File](#)

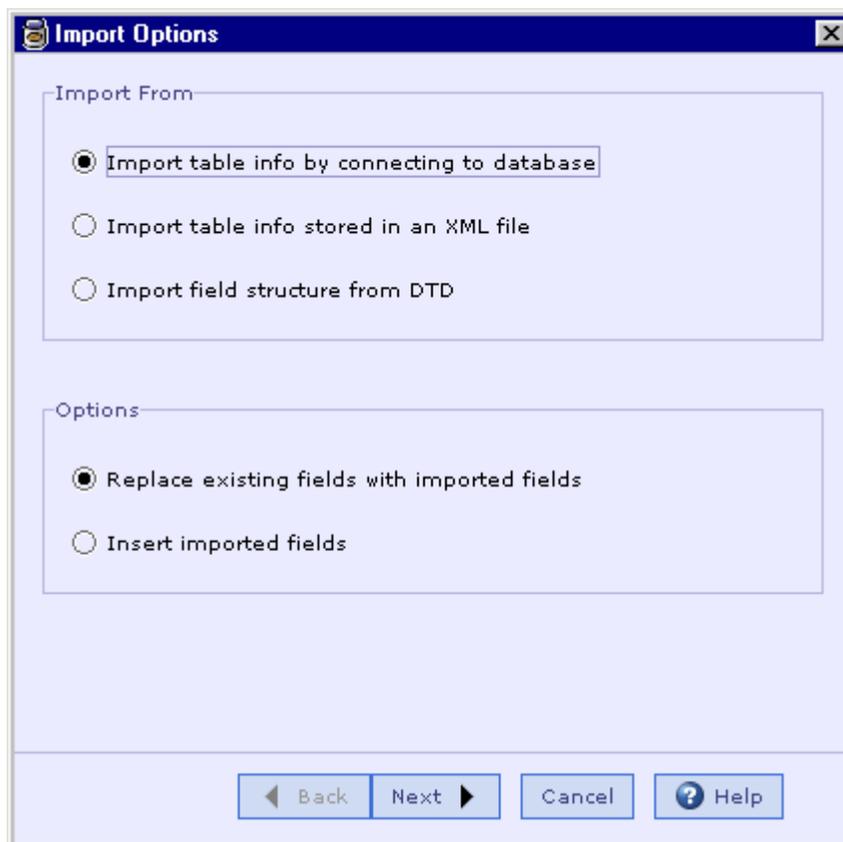
[Import Field Structure from a DTD File](#)

Import Field Structure From Database Tables

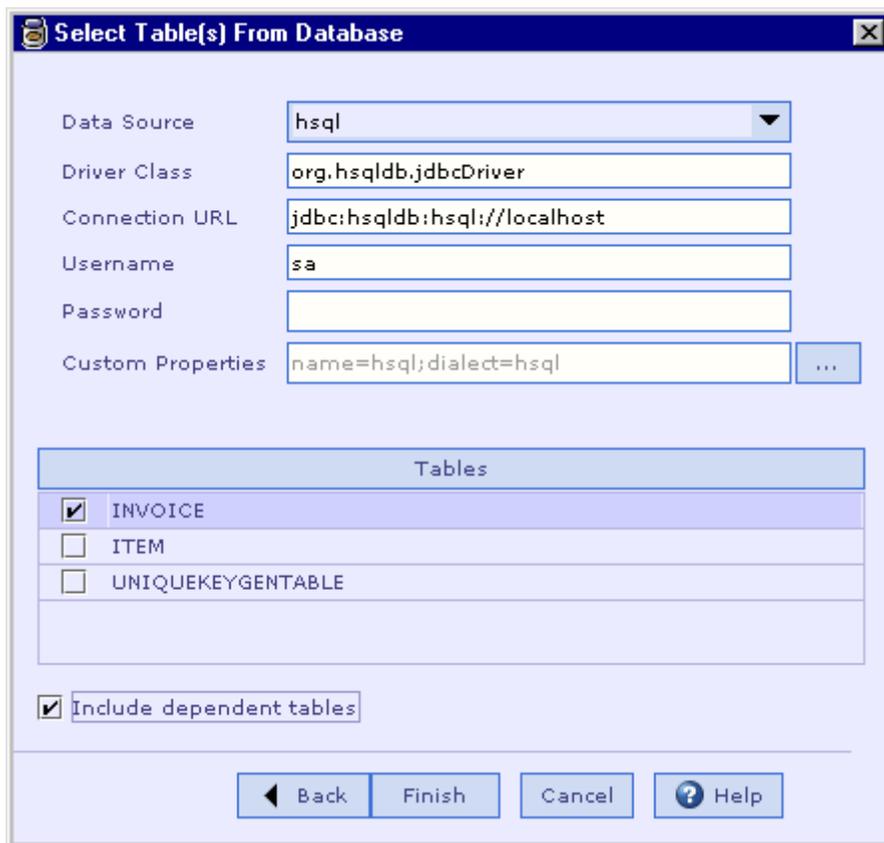
Follow the steps given below to import the internal format structure from tables of the specified data source.

1. Click on the **Import Field Structure** button in the toolbar at the top of the Internal Format UI.

The **Import Options** dialog is shown.

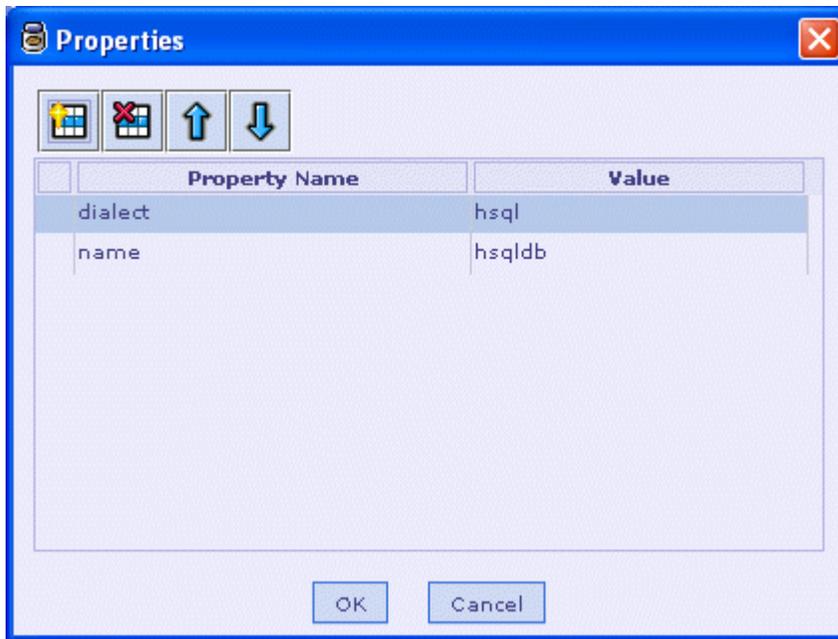


2. Select the **Import table info by connecting to database** option in the **Import From** section.
3. In the **Options** section, specify whether you want to keep or replace the existing fields and click on the **Next** button.
4. The **Select Table(s) From Database** dialog box is shown.



5. From the **Data Source** drop down list box choose the database platform to be connected.
6. In the **Driver Class**, **Connection URL**, **Username** and **Password** text boxes specify the values used in connecting to the database.

To specify database specific properties use the **Custom Properties** item. Clicking on the ellipsis button brings up a **Properties** dialog box that can be used to specify the name and value of each custom property.



7. Now click on the **Tables** button. Designer connects to the specified data source and populates tables available in that data source.
8. Select the check boxes beside the tables to be imported.
9. Select the **Include dependent tables** check box to import the tables related to the selected tables.
10. Click on the **Finish** button to start importing the field structure.

See Also:

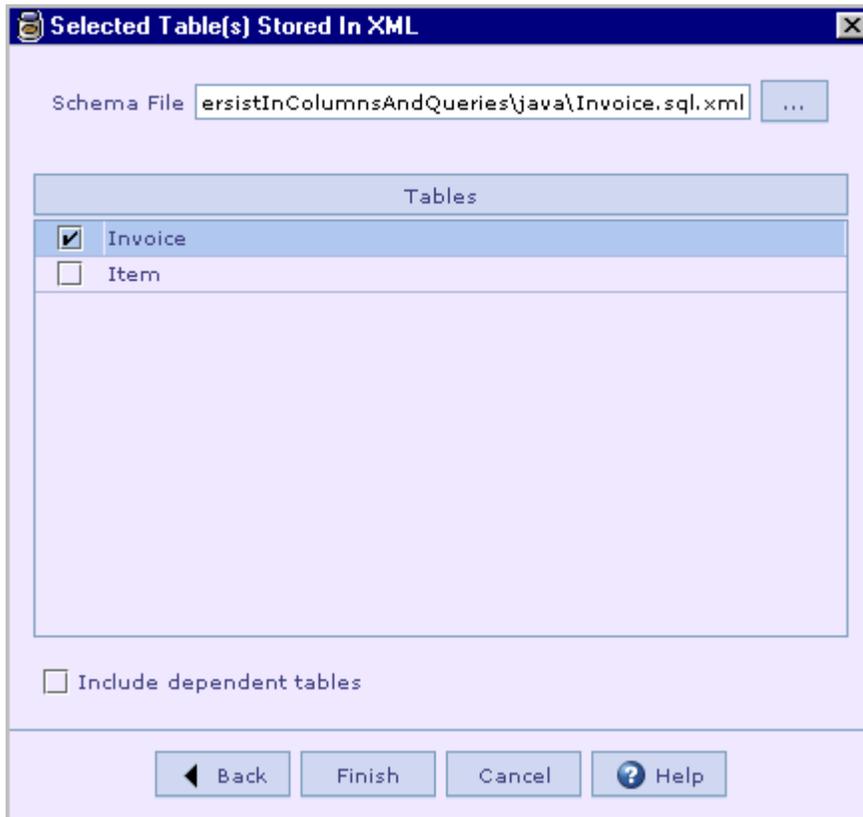
[Importing Field Structure from External Sources](#)

Import Field Structure from an XML Schema File

Follow the steps given below to import the internal format structure from table information available in an XML schema file.

1. Click on the **Import Field Structure** button in the toolbar at the top of the Internal Format UI.
The **Import Options** dialog is shown.
2. Select the **Import table info stored in an XML file** option in the **Import From** section.
3. In the **Options** section, specify whether you want to keep or replace the existing fields and click on the **Next** button.

- The **Select Tables Stored in XML** dialog is shown.



- In the **Schema File** text box, specify the XML file from which the field structure needs to be imported. Clicking on the ellipsis button besides the **Schema File** text box brings up the **Open** dialog box that can be used to select the required XML schema file.
- Now click on the **Tables** button. Designer populates table information available in the XML schema file.
- Select the check boxes beside the tables to be imported.
- Select the **Include dependent tables** check box to import the tables related to the selected tables.
- Click on the **Finish** button to start importing the field structure.

See Also:

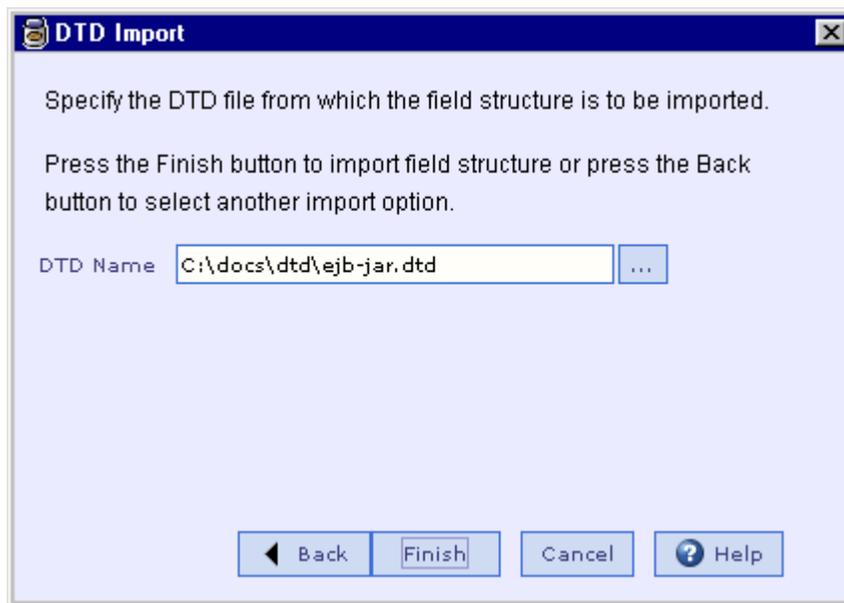
[Importing Field Structure from External Sources](#)

Import Field Structure from a DTD File

Follow the steps given below to import the internal format structure from a DTD file.

1. Click on the **Import Field Structure** button in the toolbar at the top of the Internal Format UI.
The **Import Options** dialog is shown.
2. Select the **Import field structure from DTD** option in the **Import From** section.
3. In the **Options** section, specify whether you want to keep or replace the existing fields and click on the **Next** button.

The **DTD Import** dialog box is shown.



4. In the **DTD Name** text box, specify the name of the DTD file from which the field structure needs to be imported. Clicking the ellipsis button beside the **DTD Name** text box brings up the **Open** dialog box that can be used to select the required DTD file.
5. Click on the **Finish** button to start importing the field structure.

See Also:

[Importing Field Structure from External Sources](#)

Queries

The 'Queries' child node of the 'Persistence Designer' node can be used to define a set of queries for the corresponding internal message. These queries can be invoked at runtime to retrieve internal messages that match the query criteria.

See Also:

[Defining Queries In Persistence Designer](#)

[Invoking Queries From Simulator](#)

[Removing Queries](#)

[Database Table Design](#)

[Database Mapping](#)

Defining Queries In Persistence Designer



Select the query node and define your queries.

1. To add a new query, click the New Query tool button. The New Query dialog is displayed as shown below.
2. Enter the name of the query and then add the parameters for the query (if any).
3. Enter the SQL statement. In the SQL statement you can use the parameters you have defined with \$ prefix. In this version, formulae are not supported; you have to type in the SQL statement for your query. You need be careful about the query strings, since the designer does not validate them.

Instead of entering an SQL statement directly, you can enter a formula in the Formula text box and click the "Generate Query" button to generate the SQL statement. Note that only a limited set of formula functions can be used here. See the figure given below.

Edit Query

Query Name:

Parameters

Parameter Name	Data Type
◆ Id	String

Query Condition

Enter the SQL query. Alternatively, enter a formula in the 'Formula Text Field' and click 'Generate Query' to view the SQL generated.

Formula

SQL

```
SELECT DISTINCT Invoice.Invoice_PK, Invoice.InvoiceDate,
Invoice.ClientID FROM Invoice Invoice, Item where
Invoice.Invoice_PK = Item.Invoice_FK and Item.ItemID = $Id
```

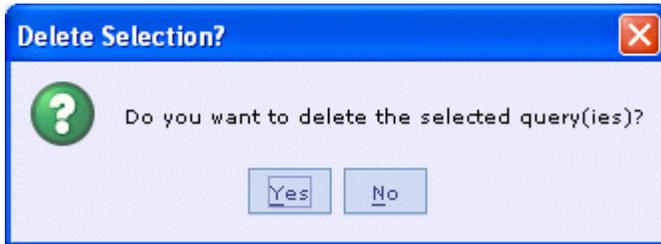
See Also:

[Invoking Queries From Simulator](#)

[Removing Queries](#)

Removing Queries

In the Queries UI select the query to be removed. Click the  button. The 'Delete Selection' dialog is displayed.



Click 'Yes' to delete the selected query(ies).

See Also:

[Defining Queries In Persistence Designer](#)

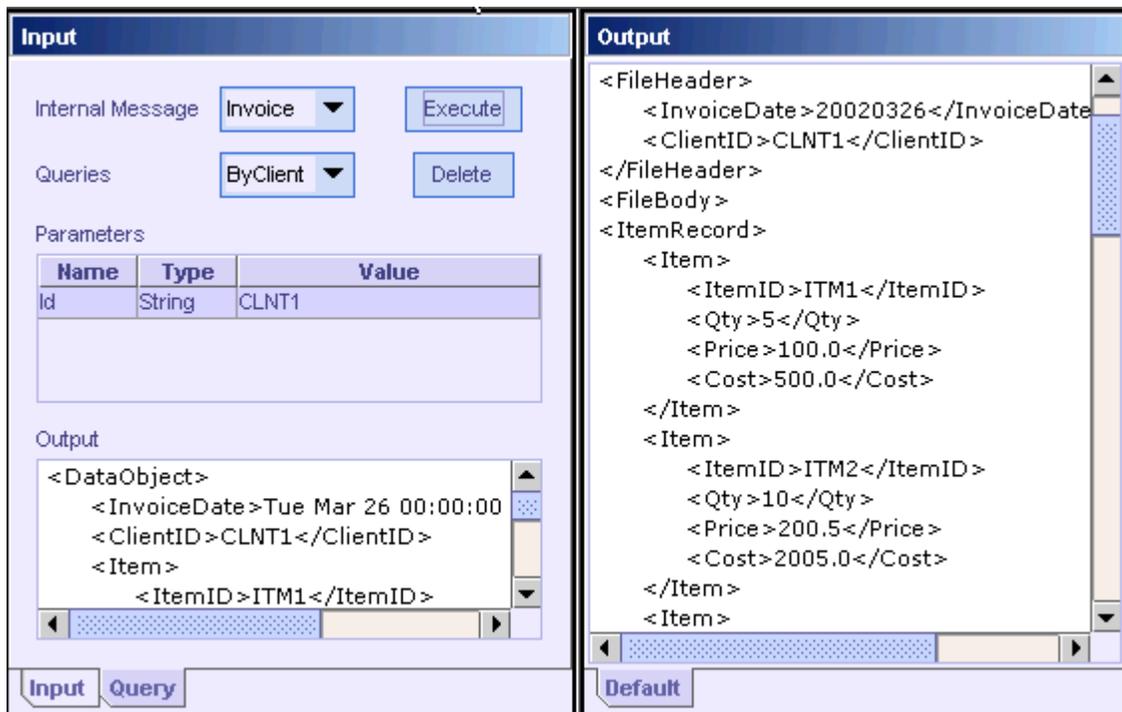
[Invoking Queries From Simulator](#)

Invoking Queries From Simulator

The Query tab in the input pane of Simulator can be used to invoke queries defined in the currently deployed cartridge as described below:

1. Select the query to be executed from the Queries combo box.
2. Specify parameter values, if any.
3. Click on the Execute button to execute the query.

The query result is displayed in the Output text area. See the figure given below.



Similarly, records matching a query can be deleted by first selecting a query and then clicking the Delete button.

See Also:

[Defining Queries In Persistence Designer](#)
[Removing Queries](#)

Database Mapping

Database mapping can be defined by specifying mapping rules. There are two types of mapping rules as given below:

One-to-One Mapping Rule- In this, a target field/section is mapped to a source field/section.

Formula Mapping Rule- In this, a target field/section is assigned a formula involving one or more source fields/sections. The formula can also involve fields/sections from the target message format.

When persistence designer is created, the internal message fields are automatically mapped to the fields in the persistence designer as one-to-one mapping rule.

See the section Mapping Rules for information on field/section mapping rules.

See Also:[OR Mapping Rules and Behavior](#)[Database Table Design](#)[Queries](#)

Updating Persistence Designer from Internal Message

After a persistence designer has been created, the internal message based on which it was created may be changed. In such cases the persistence designer should also be updated. There are two ways of updating persistence designer,

Reinitializing Persistence Designer - in this case the changes you have made to it are lost and the entire Persistence Designer is recreated from the internal message.

Updating Persistence Designer - in this case only the changes that were made to the internal message are updated in the Persistence Designer. The changes you made in the Persistence Designer are not lost.

See Also:[Reinitializing Persistence Designer](#)[Updating Persistence Designer](#)[Adding persistence support](#)[Removing Persistence Designer](#)

Reinitializing Persistence Designer

Right click the Persistence Designer and select **Reinitialize from Internal Message** menu item from the context menu. Persistence Designer will be reinitialized. Any settings you have made up then in the Persistence Designer will be lost. In order to avoid it you can use **Update from Internal Message** menu item.

See Also:[Updating Persistence Designer](#)

Updating Persistence Designer

Right click the Persistence Designer and select Update from Internal Message menu item from the context menu. If any fields have been added to/removed from the internal message only those fields will be added to/removed from the internal message. The settings of other fields will not be lost.

Note:

If a field name is changed and **Update from Internal Message** is selected the field's setting stored previously will be lost.

See Also:

[Reinitializing Persistence Designer](#)

Removing Persistence Designer

You can remove the persistence designer by right clicking it and selecting the 'Delete' menu item. You can also remove it by right clicking the internal message and selecting 'Remove Persistence Designer' menu item.

See Also:

[Adding persistence support](#)

[Updating Persistence Designer From Internal Message](#)

[Persistence Designer](#)

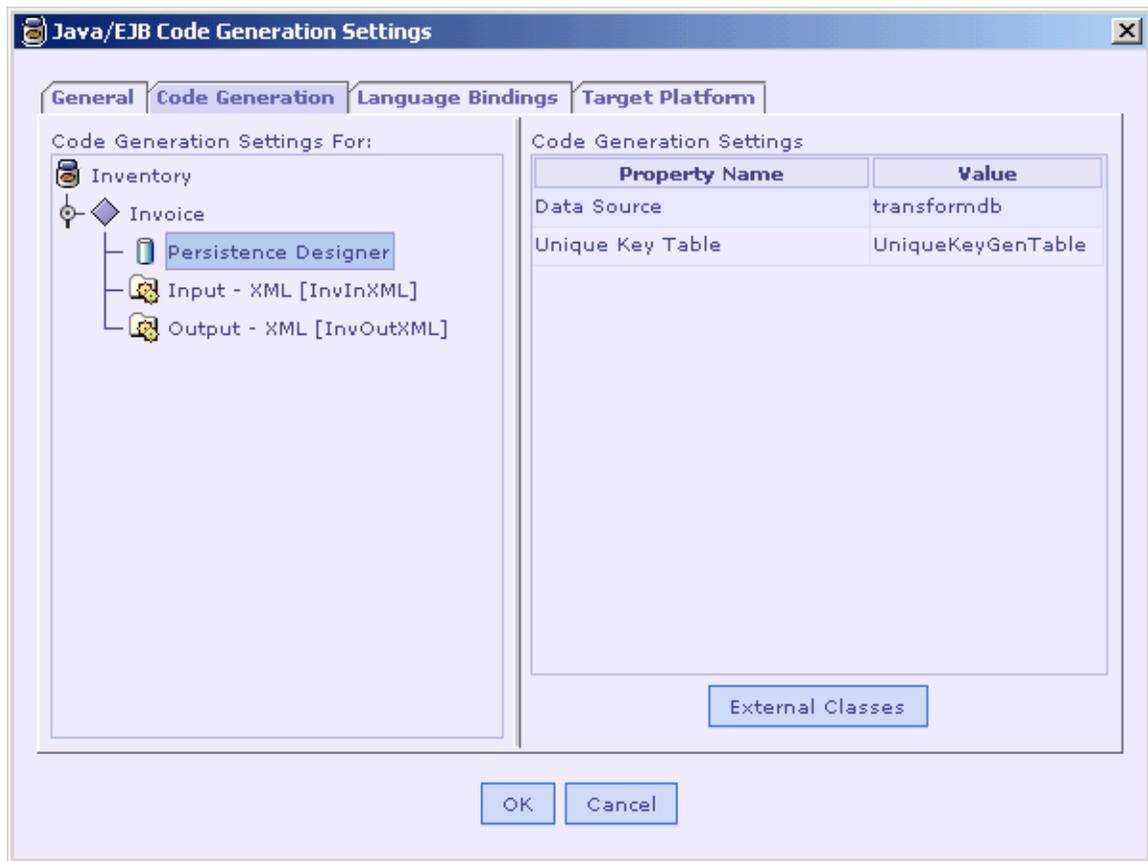
Configuring the Data Source

The connection string to be used at runtime should be configured in the server. From the designer you can configure the name of the data source to be used at runtime. Corresponding to this name, you should define data source in the server (how you do this is of course specific to the server).

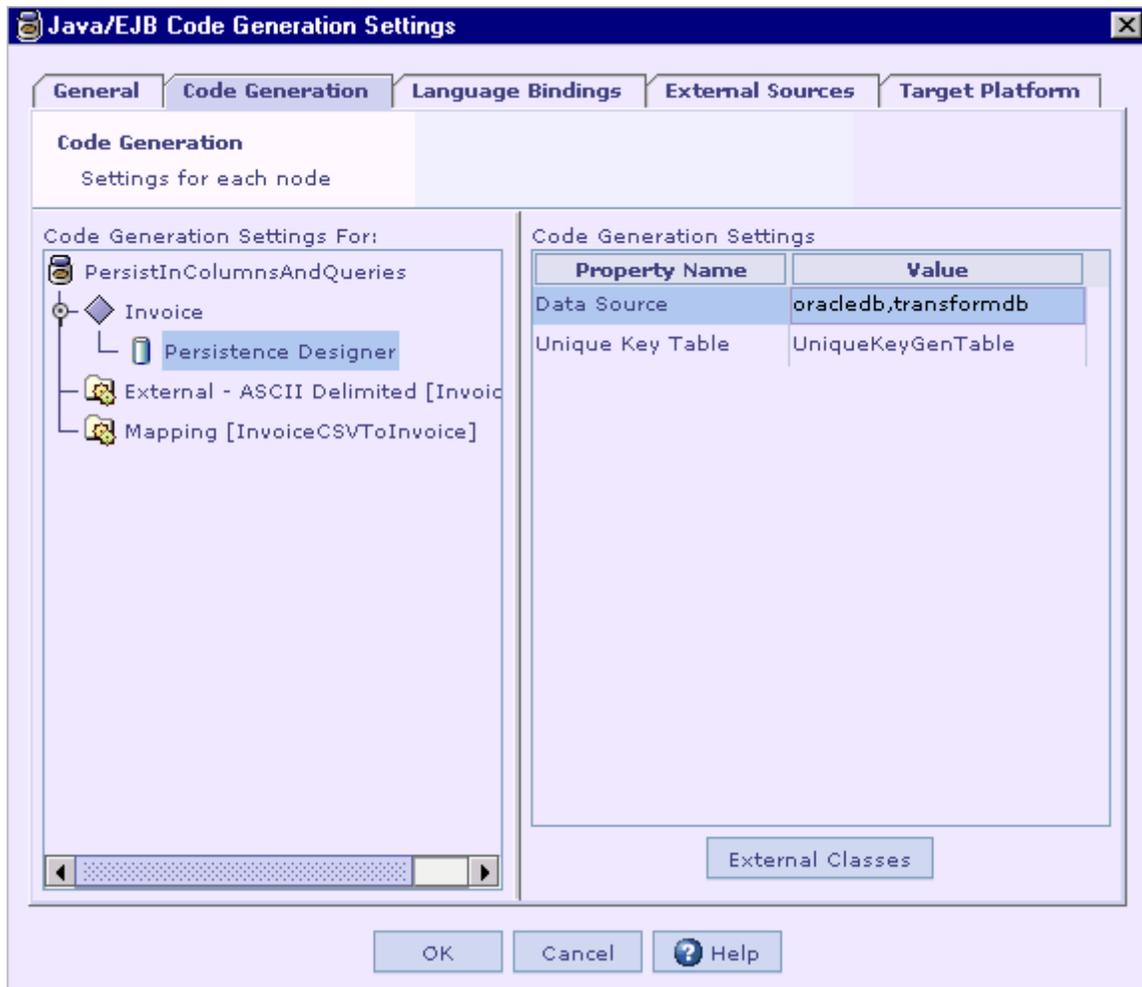
To configure the data source,

1. Select **Code Generation settings** menu. In the dialog select Persistence Designer you want to configure.

2. Enter the name of the data source.



3. Multiple data sources could be associated with the Persistent Entity. The data source names could be specified as comma separated values.



- The Data Source that is specified first is the Default Data source. For example in the above diagram 'oracledb' is the default data source.

See Also:

[Populating tables](#)

[Persistence Designer](#)

Populating Tables

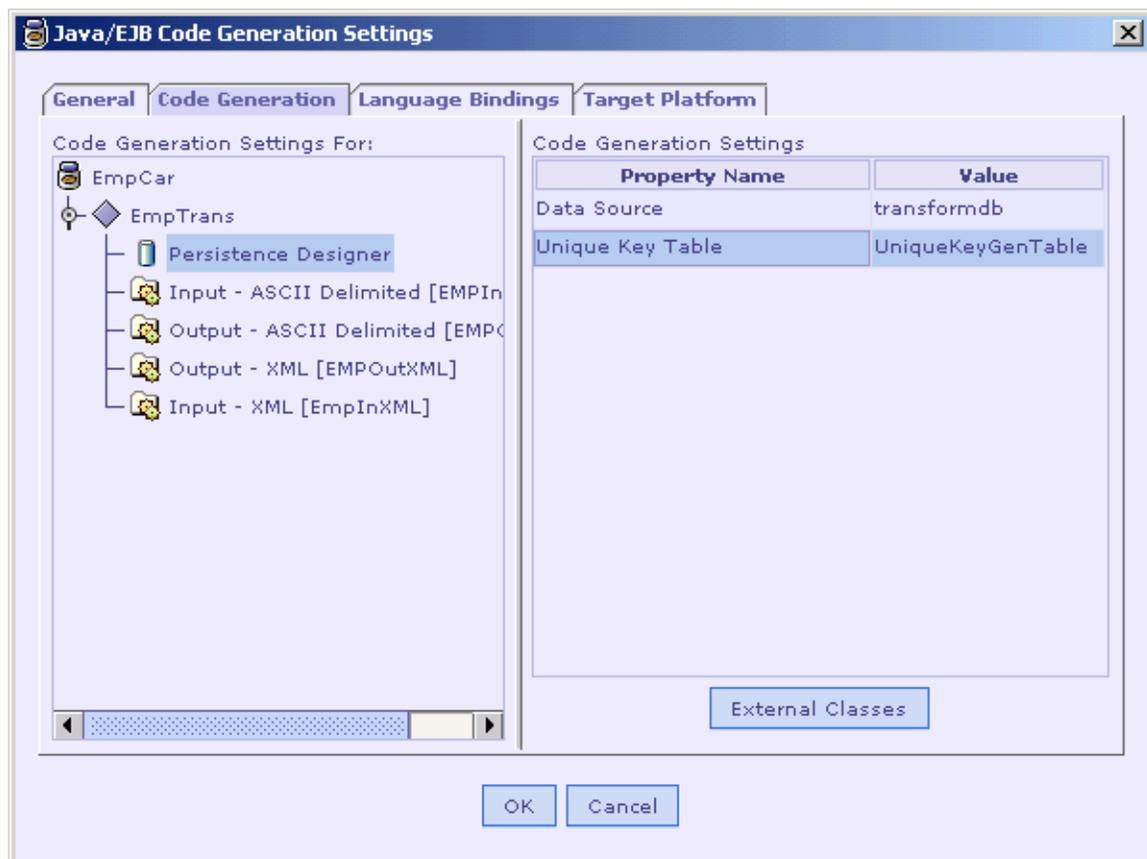
During code generation an XML file that contains the schema of the tables to be created in the database is generated (in the `cartridgedir\Java` directory). You can use the Execute SQL menu item under the Tools menu of Designer to populate the tables from the XML schema. Alternatively, you can convert this XML to the SQL file for your database using the SQL Generator tool (under the Tools menu).

Note:

The Persistence Designer uses a unique key generator based on a database table. Make sure that you create this table in the database you are using.

```
drop table UniqueKeyGenTable;
create table UniqueKeyGenTable(CurrentKey BIGINT);
insert into UniqueKeyGenTable values(100);
```

You can also change the name of the unique key generation table in the code generation settings of the persistence dialog as shown below.



By default the unique key table will be 'UniqueKeyGenTable'.

If you change the name of the unique key table please make sure that you create a table of the same name.

If there are multiple users accessing the same table using the same 'UniqueKeyTable', and one user changes the unique key table in his cartridge using code generation settings and other users do not change the unique key table in their cartridges error may occur. This is because since different unique key tables are now used, the same key may be generated twice.

In a case where two applications write data to the same table and both of them use the same 'UniqueKeyTable' there will be no problem. But if they use different 'UniqueKeyTables' then error may occur. This is because since different unique key tables are now used the same key may be generated twice.

See Also:

[Configuring the Data source](#)

[Runtime configuration](#)

[Persistence Designer](#)

Runtime configuration

Before you run the server, start the database server.

Populate the table structure using the schema file produced during code generation. Refer to DB tools documentation.

Configure the datasource. The name of the data source should be same as that you specified in the code generation settings for the Persistence Designer.

Add the following to data-source.xml in the server config directory. Change connection parameters to match your database.

Copy the JDBC driver to server\lib directory

```
<data-source
  class="com.evermind.sql.DriverManagerDataSource"
  name="MYSQL"
  location="jdbc/MYSQLDS"
  xa-location="jdbc/xa/MYSQLXADS"
  ejb-location="transformdb"
  connection-driver="com.mysql.jdbc.Driver"
  username=""
  password=""
  url="jdbc:mysql:///test"
```

```
    inactivity-timeout="30"  
  />
```

See Also:[Populating tables](#)[OR Mapping Rules and Behavior](#)[Persistence Designer](#)

OR Mapping Rules and Behavior

General Restrictions

All non-nullable DB fields except auto generated fields and foreign keys should necessarily have a mapping.

All tables except the leaf tables should necessarily have a primary key.

Each child table should have a foreign key referring to its immediate parent's primary key.

Auto Generated Keys

If the auto generated DB field has a mapping

When inserting the record, a new value is auto generated and used (instead of the mapped NO field's value). The generated value is assigned back to the NO field.

When selecting a record the NO's field is populated using the DB value.

If the auto generated DB field has no mapping,

A field in NO is synthesized. When inserting the record, a new value is auto generated and used. The synthesized field is filled using the auto generated value.

When selecting the record the NO's synthesized field is populated using the DB value.

Foreign Keys

If the **foreign key** DB field has mapping,

When inserting the record, the value is correctly populated from the parent object (to which it refers). The value is also assigned back to mapped NO field.

When selecting the record the **foreign key** DB field is assigned to the mapped NO field.

If the foreign key DB field has no mapping,

When inserting the record, the value is correctly populated from the parent object (to which it refers).

When selecting the record the **foreign key** DB field is ignored.

See Also:

[Runtime configuration](#)

[Persistence Designer](#)