Getting Started

IBM Business Process Manager

Query Table Builder
SupportPac PA71

Version 3.0
June, 2011

Table of contents

Introduction

This document describes how to install and use the Query Table Builder.

- Chapter 1 specifies the requirements of the Query Table Builder, and how to install it on a suitable development environment.
- Chapter 2 provides step-by-step instructions for creating a query table definition file, adding contents to this query table, and using the test client to deploy and test the new query table.
- Chapter 3 describes the usage of the Query Table Builder in detail.
- Chapter 4 explains how to export a query table definition so that it can be deployed on a production system of IBM Business Process Manager 7.5.
- Chapter 5 assists you in finding solutions to the most common issues that arise when using the Query Table Builder.

Note: For a description of query tables, refer to the following topics in the Business Process Management Information Center:

Concepts:
http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=/com.ibm.wbm.bpc.doc/topics/c6bpel_querytables.html

Administration:
http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=/com.ibm.wbm.bpc.doc/topics/t4querytables_admin.html

Change History

Changes from V1.0 to V1.1

- Increased minimum requirements for Eclipse, GEF, EMF and WebSphere Integration Developer in chapter 1.1 (System and software requirements).
- Added update instructions in chapter 1.2 (Installing or updating the software).
- Minor updates in chapter 2 (Getting started) to reflect version changes.
- Added descriptions for new features in chapter 3 (Using the Query Table Builder):
  3.1.1 Query Tables perspective
  3.1.8 Localization
  3.2.3 Creating, modifying and deleting a language
  3.2.13 Creating a composite query table for Business Space for Human Workflow
- Updated chapter 4 (Exporting and importing query table definitions) to describe the new Query Table export wizard.
- Added chapter 5 (Troubleshooting).
- Updated screenshots in the whole document.

Changes from V1.1 to V1.2

- Added version overview table in chapter 1.3 (Verifying the installation).
- Updated instructions in chapter 3.2.13 (Creating a composite query table for Business Space for Human Workflow).

Changes from V1.2 to V2.0

- Use WebSphere Integration Developer 7.0 and WebSphere Process Server 7.0
- Updates in chapter 3.2.13 – new type of composite query tables for Business Space widget Escalations List.

Changes from V2.0 to V2.1

- Updates in chapter 1.1.2 – added description of compatibility issues between Query Table Builder version 2.1 and WebSphere Integration Developer Version 7.0.0.0 and 7.0.0.1.
- Updates in chapter 3.2.13 – new type of composite query tables for Business Space widget Work Baskets List, introduced in WebSphere Process Server 7.0 Feature Pack 1.
- Updates in chapter 3.4 – new option in the test client that allows to perform a query as a system administrator.
- Updated screenshots in various sections of the document.

Changes from V2.1 to V3.0

- Updates to reflect new product IBM Business Process Manager 7.5

1. Installing the Query Table Builder

This section describes how to prepare for and install the Query Table Builder.

1.1. System and software requirements

To use the Query Table Builder, you need a suitable development environment.

1.1.1. Operating system

Choose one of the following operating systems:

- Windows® XP
- Windows Vista™
- Linux® (tested on Red Hat Linux 4.7)

1.1.2. Integrated Development Environment

An Eclipse Integrated Development Environment (IDE) consisting of:

- Eclipse 3.3, 3.4 or higher
- Eclipse Modeling Framework (EMF) 2.3 or 2.4
- Graphical Editing Framework (GEF) 2.3 or 2.4

Instead of downloading each component individually from http://www.eclipse.org, it is recommended that you use an Eclipse distribution that at least contains the above listed components.

We recommend to download one of the following packages from http://www.eclipse.org/downloads, which already contain all necessary components:

- Eclipse IDE for Java EE Developers
- Eclipse IDE for Java Developers
- Eclipse Modeling Tools
- Eclipse for RCP/Plug-in Developers
- Eclipse IDE for Java and Report Developers

If you plan to use Integration Developer for the Query Table Builder, you can only use Integration Developer Version 7.5. Previous releases of Integration Developer will fail if you install the Query Table Builder.

1.2. Installing or updating the software

Download the file **pa71.zip** into a temporary directory on your computer and extract the file into this directory. Make a note of the location of this directory.

Notes: The installation instructions and dialogs differ slightly between Eclipse 3.3 and Eclipse 3.4 and newer Integrated Development Environments. Steps that need to be executed differently are marked with *Eclipse 3.3* and *Eclipse 3.4+*. All other steps are common. The screenshots in this section are taken from the Eclipse 3.4 package "Eclipse Modeling Tools". Integration Developer Version 7.5 is based on Eclipse 3.6, therefore you should follow the Eclipse 3.4+ instructions.

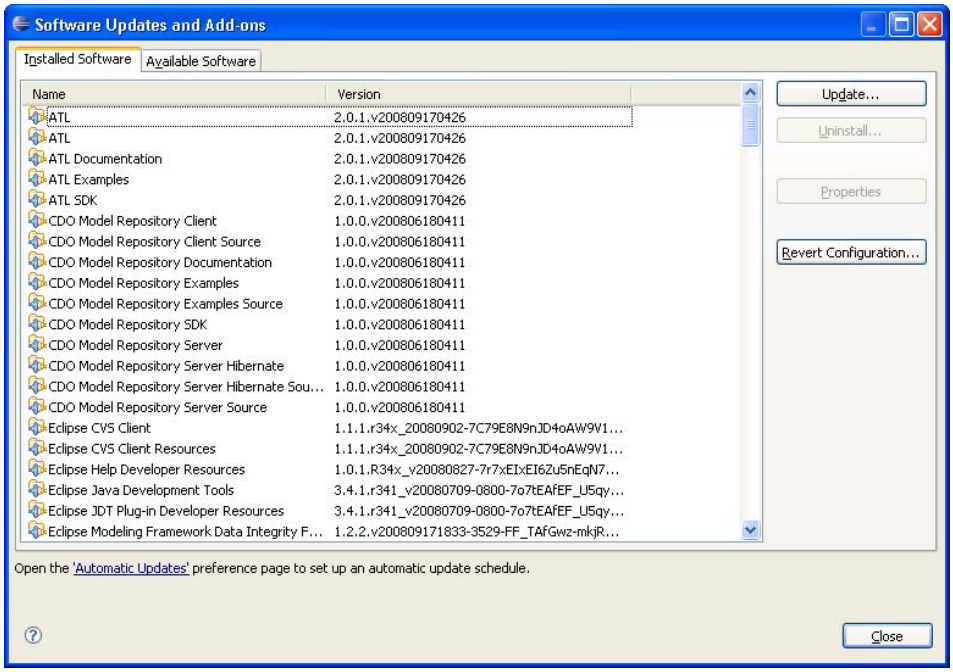
1. Start your Integrated Development Environment. For installation, enter a new workspace or use an existing one.

2. In the Eclipse menu, select

   Eclipse 3.3: **Help > Software Updates > Find and Install**

   Eclipse 3.4+: **Help > Software Updates**

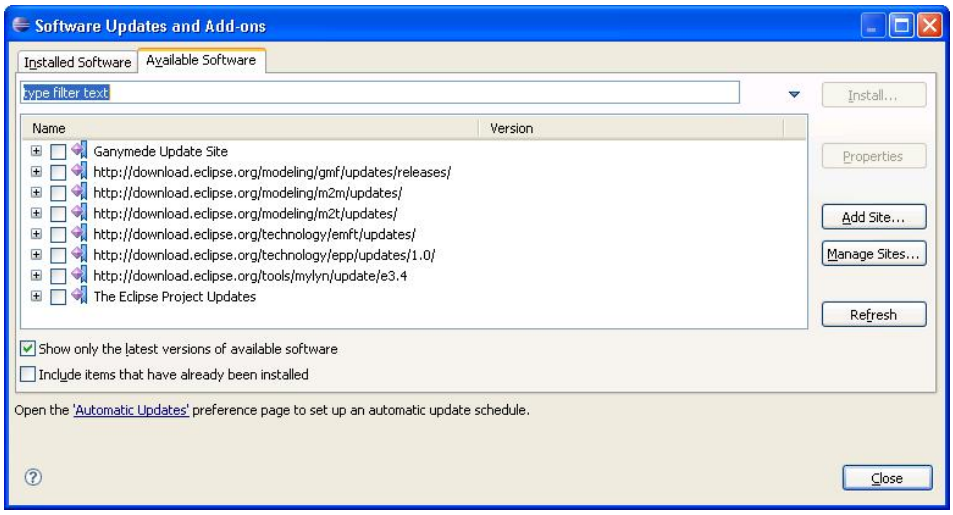
   If your Integrated Development Environment does not offer this menu, switch to the resource perspective using the Eclipse menu **Window > Open Perspective > Other**. In the dialog, select **Resource**, and click **OK**.



3. In the dialog that appears:

   Eclipse 3.3: Select **Search for new features to install** and click **Next**.

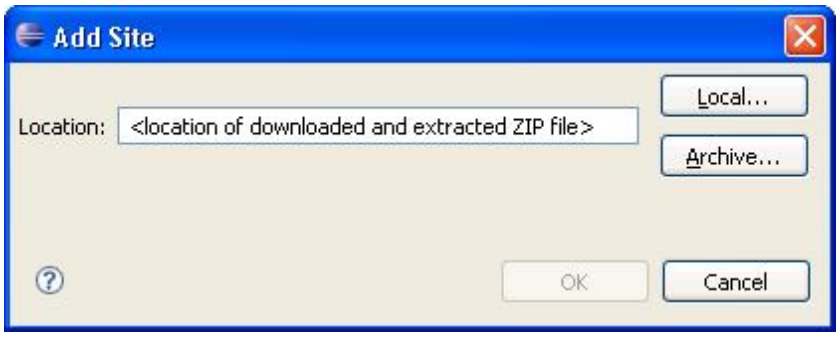
   Eclipse 3.4+: Select the **Available Software** tab.



4. *Eclipse 3.3:* Click **New Archived Site**.

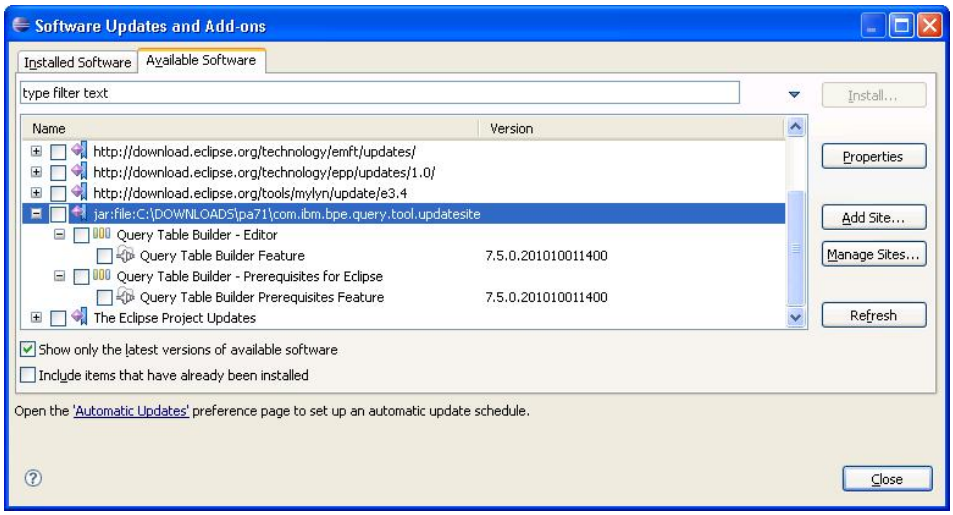
   Eclipse 3.4+: Click **Add Site** and then ***Archive***.

   Browse to the location of the temporary directory where you extracted the downloaded .zip file to. Select the file com.ibm.bpe.query.tool.updatesite.zip. Click **Open** and **OK** to add the new site to the list of update sites.



5. *Eclipse 3.3:* Make sure that the added site is selected, and click **Finish**. In the following dialog, expand all subtrees.

   Eclipse 3.4+: The added site is automatically expanded.



6. Select from the following features:
   - **Query Table Builder Feature:** Always select this feature.
   - **Query Table Builder Prerequisites Feature:**
     Select this feature only if you are installing the Query Table Builder in an Integrated Development Environment that is not Integration Developer Version 7.5.

     Notes: If you are installing the Query Table Builder in Integration Developer Version 7.5, do **not** check this feature. Installing this feature may restrict the functionality of Integration Developer.

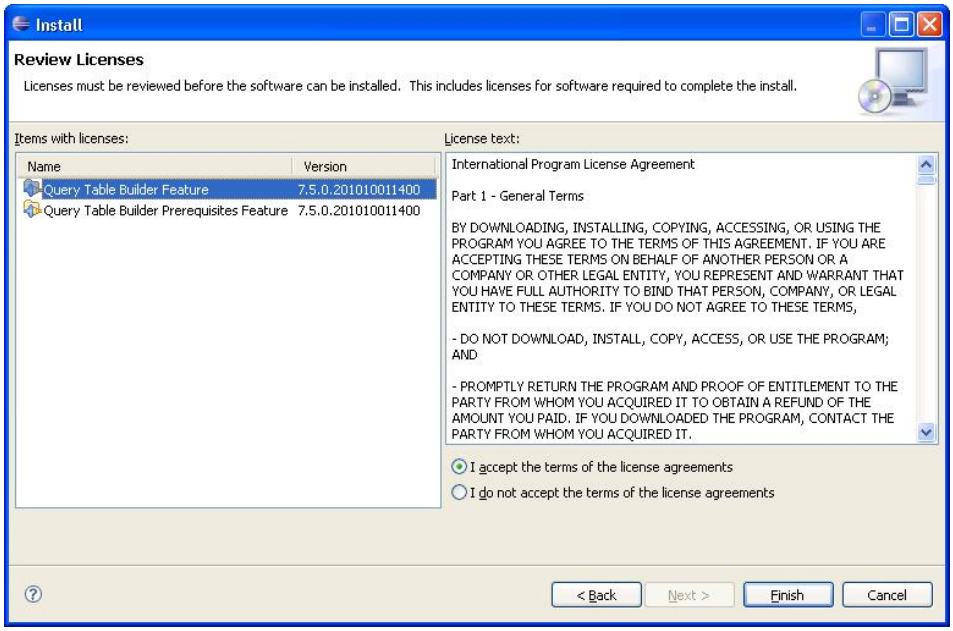
   Eclipse 3.3: Click **Next**.

   Eclipse 3.4+: Click **Install** and then **Next**.

7. Review the license text for the Query Table Builder and select 'I accept the terms of the license agreements'.

   Eclipse 3.3: Click **Next** and then **Finish**.

   Eclipse 3.4+: Click **Finish**.



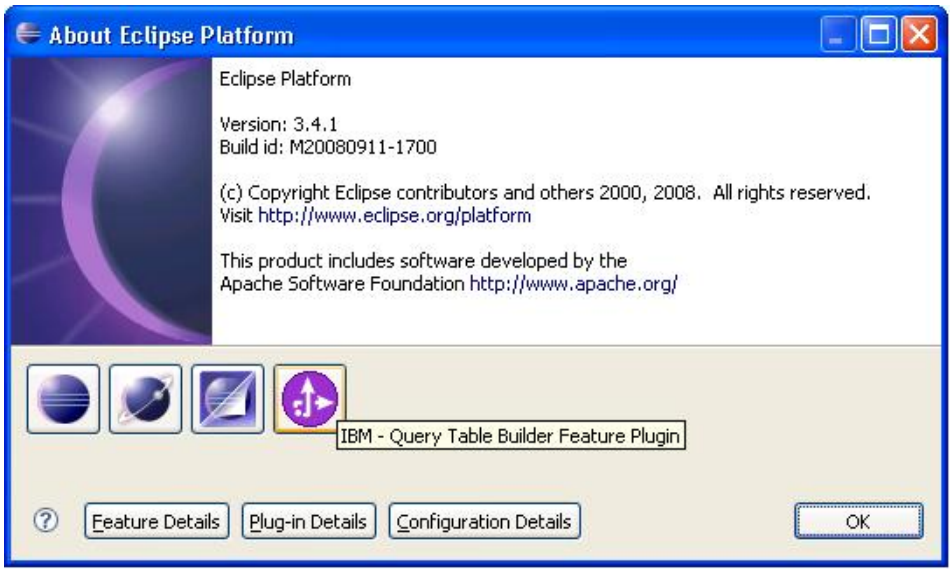
8. The Query Table Builder is not digitally signed, therefore you might get a verification dialog for installing an unsigned feature. Click **Install** or **Install All** to verify that you trust this feature.

9. After the installation is finished, you are prompted to restart your Eclipse based Integrated Development Environment. Click **Yes** to continue. This step finishes the installation of the Query Table Builder.

1.3. Verifying the installation

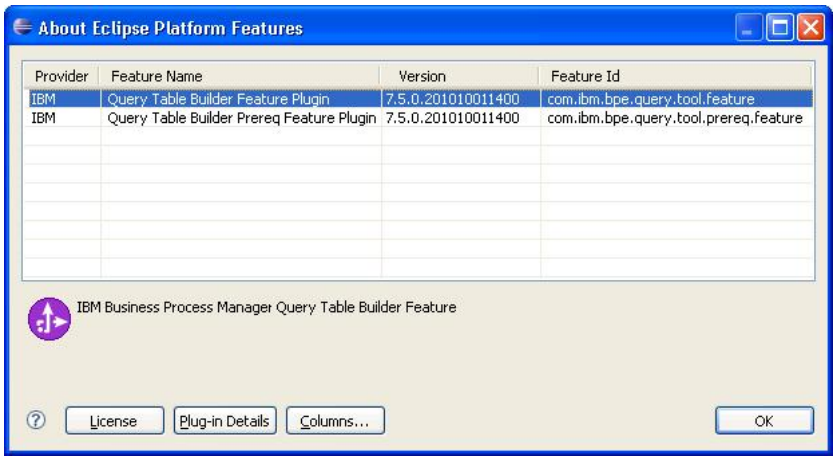
To verify your installation, complete the following steps.

1. Click **Help > About Eclipse Platform**. Depending on your Integrated Development Environment, this menu item might have a slightly different name.



2. Click on the icon  .

3. The following dialog opens. You may review the license or plug-in details of the features you installed.



If you need to find out which version of the Query Table Builder you have installed, you may compare the version of the feature 'Query Table Builder Plugin' with the following table:

Feature version	Version	Released	Main content
6.3.0. 200812051700	1.0	12/2008	Initial Release
6.3.1. 200812051700	1.0 (update)	01/2009	Bug fixes
6.3.100. 200812051700	1.1	04/2009	Internationalization Support
6.3.101. 200812051700	1.2	07/2009	Wizards for WebSphere Process Server Feature Pack
7.0.0. 200812051700	2.0	12/2009	WID v7.0 support Update to Wizards for Query Tables for Business Space
7.0.200. 200812051700	2.1	06/2010	Wizard for Business Space widget Work Baskets List, introduced in WebSphere Process Server 7.0 Feature Pack 1
7.5.0. 201010011400	3.0	06/2011	Updates for Business Process Manager

1.4. Uninstalling the Query Table Builder

To remove the Query Table Builder from your Integrated Development Environment, complete the following steps.

1. Start your Integrated Development Environment. For uninstallation, enter a new workspace or use an existing one.

2. In the Eclipse menu, select:

   Eclipse 3.3: **Help > Software Updates > Manage Configuration**

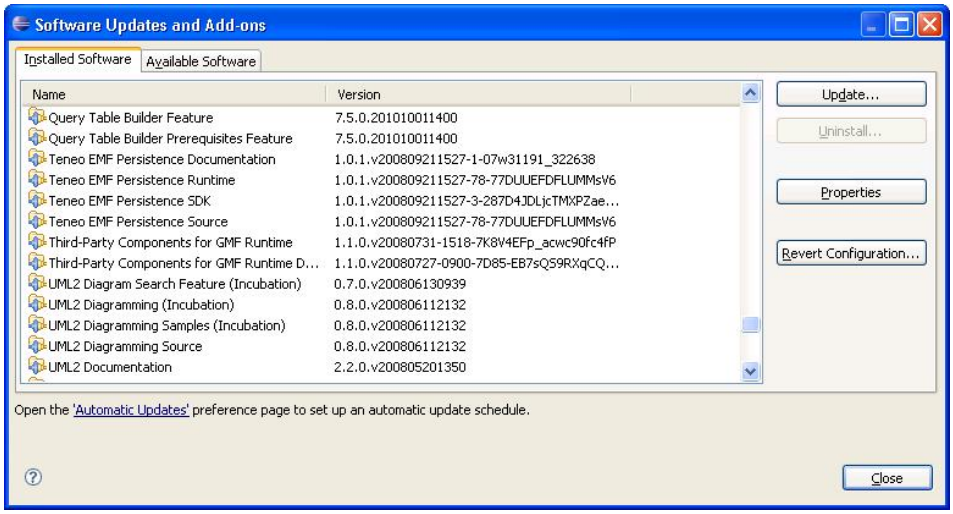
   Eclipse 3.4+: **Help > Software Updates**

   If your Integrated Development Environment does not offer this menu, switch to the resource perspective using the Eclipse menu **Window > Open Perspective > Other.** In the dialog, select **Resource** and click **OK**.

3. In the dialog that appears:

   Eclipse 3.3: Expand the tree so you can see all of the features that are installed.

   Eclipse 3.4+: Select the **Installed Software** tab.

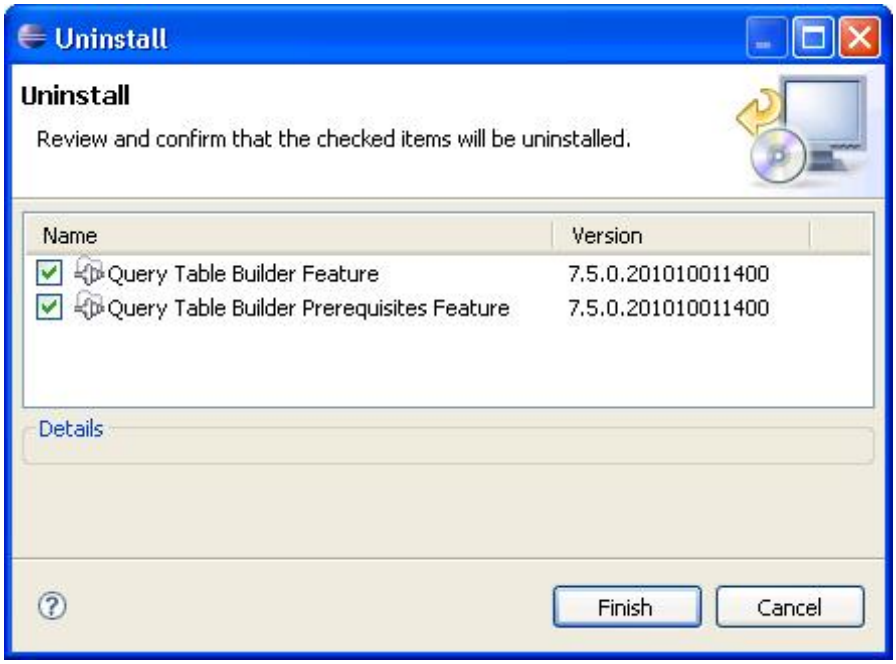


4. In the list, select the features that you installed, i.e. "Query Table Builder Feature" and "Query Table Builder Prerequisites Feature" if available.

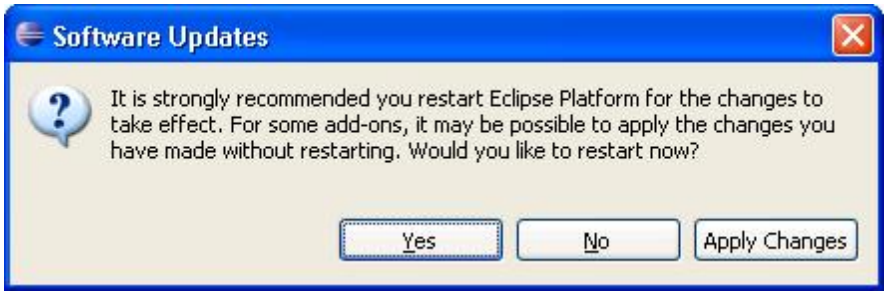
   Eclipse 3.3: Right-click on the features and select **Uninstall**.

   Eclipse 3.4+: Click **Uninstall**.

In the following dialog, click **OK** *(Eclipse 3.3)* or **Finish** *(Eclipse 3.4+)* to start uninstalling the features.



5. After the uninstallation is finished, you are prompted to restart your Eclipse based Integrated Development Environment. Click **Yes** to continue. This step finishes the uninstallation of the Query Table Builder.



2. Getting started

This section introduces you to the Query Table Builder. It provides step-by-step instructions for creating a query table definition file, adding contents to this query table, and, finally, using the test client feature of the Query Table Builder to deploy and test the new query table.

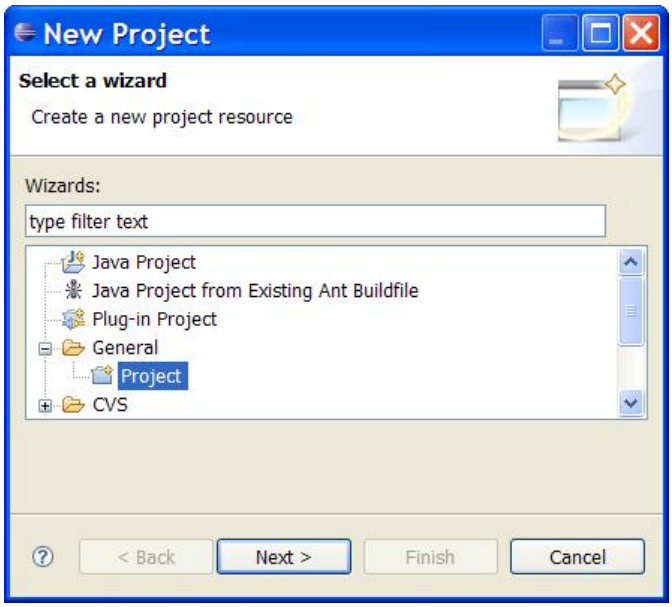
The instructions and screenshots in this section are based on Eclipse 3.4.0.

2.1. Creating a project

You can create a query table definition (.qtd) file in any type of project. In this example, a standard project is used.

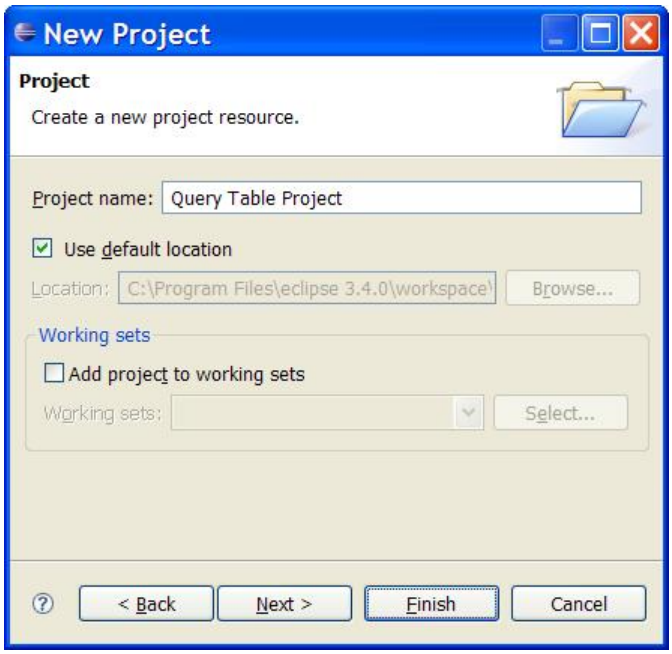
To create the project, follow these steps:

1. In the menu of Eclipse, select **File > New > Project**. The **New Project** window opens.



2. Select **General > Project** and click **Next**.

3. In the **Project name** field, enter `Query Table Project` and click **Finish** to create the project.



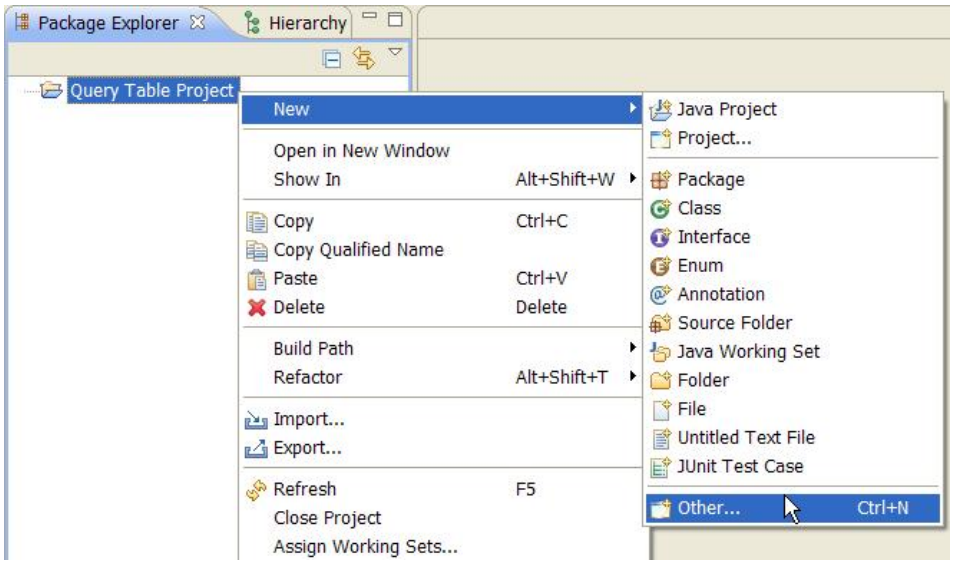
The project has been created.

2.2. Creating a query table definition file

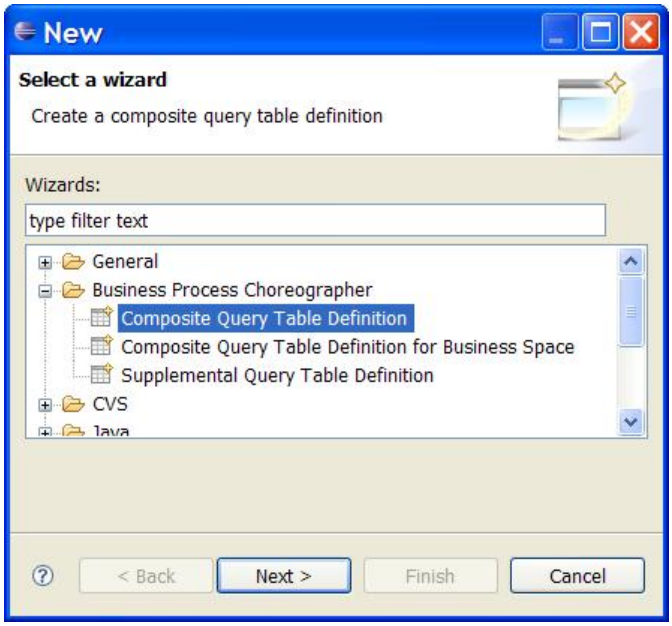
There are three different types of query tables that you can create and edit with the Query Table Builder: composite query tables (see *Working with composite query tables* on page 33), composite query tables that are intended for use in Business Space for Human Workflow (see *Creating a composite query table for Business Space for Human Workflow* on page 49), and supplemental query tables (see *Working with supplemental query tables* on page 53).

This example shows how to create a new composite query table that provides information about task instances. Perform the following steps to create a query table definition file:

1. In the navigation view on the left side, right-click the project **Query Table Project,** and select **New > Other**.

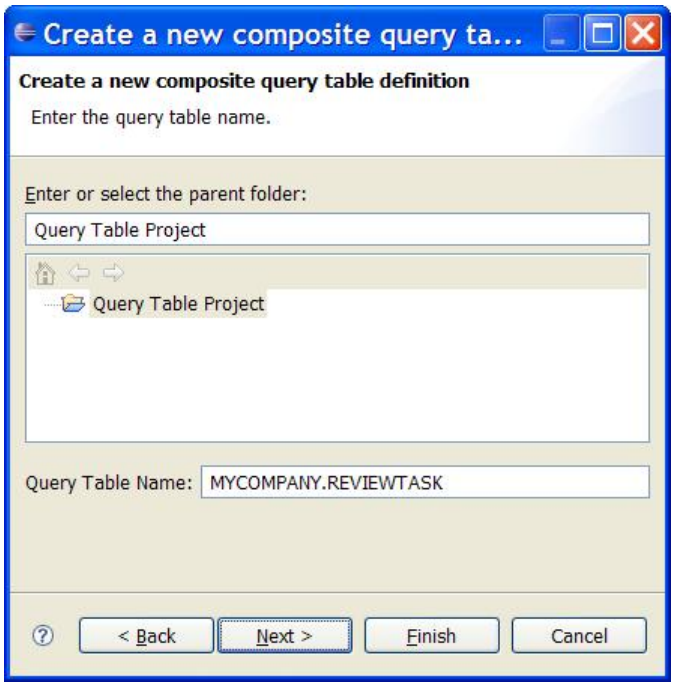


The **New** window opens.



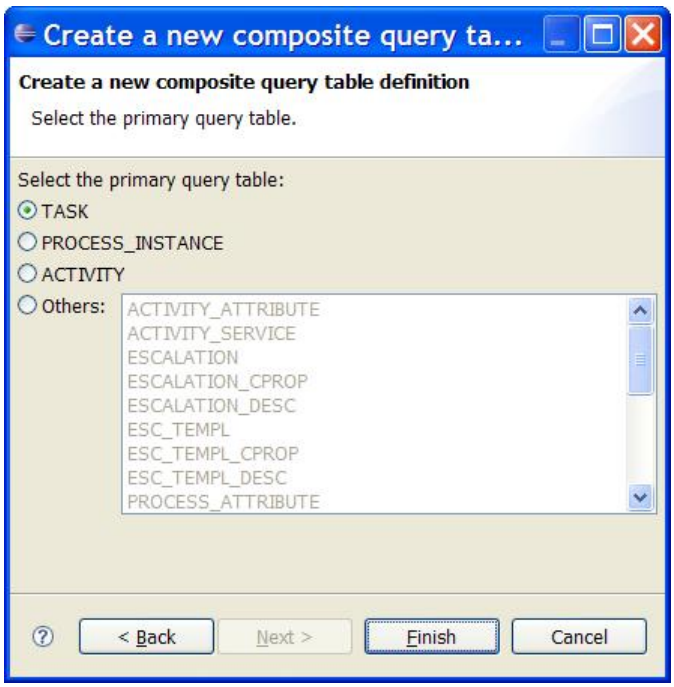
2. Select **Business Process Choreographer > Composite Query Table Definition**, and click **Next**.

3. In the **Query Table Name** field, specify `MYCOMPANY.REVIEWTASK` as the name of the new Composite Query Table.



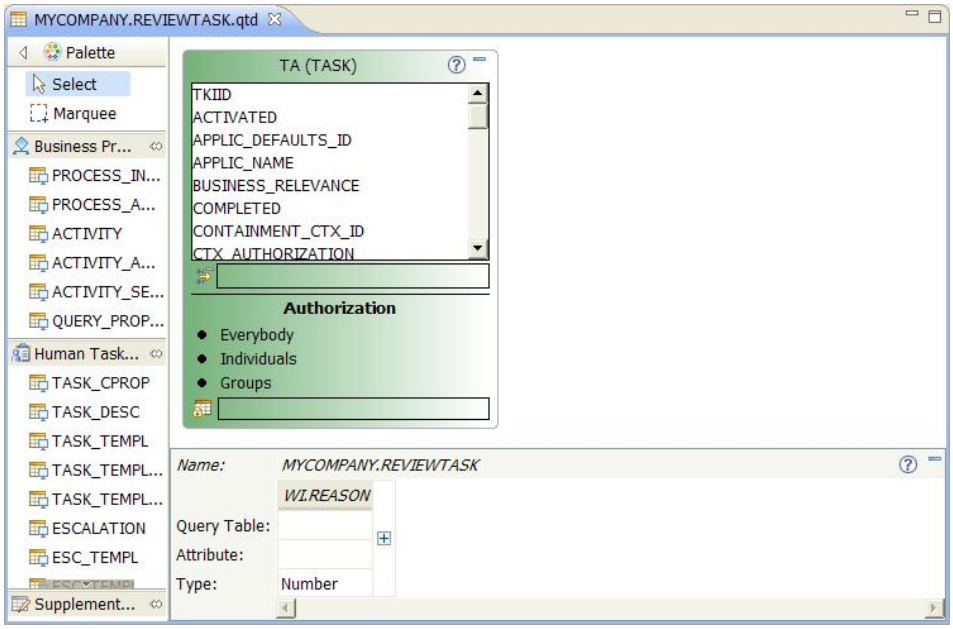
4. Click **Next**.

5. Select **TASK** as the primary query table of the new query table and click **Finish**.



6. If a dialog appears that asks you whether you want to switch to the **Query Tables perspective**, click **Yes**.

7. The new query table definition file `MYCOMPANY.REVIEWTASK.qtd` is created in the project. The file opens in the editor:



The new query table consists only of the primary query table TA. The primary query table is shown in green and is by default always in the upper left corner.

2.3. Creating the query table

In this example, you will create a query table that contains:

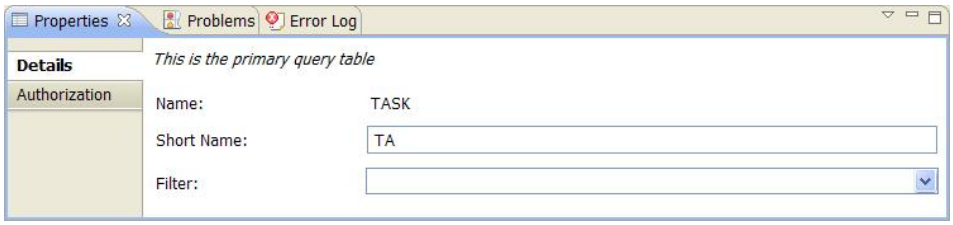
- All task instances of the template *ReviewHumanTask* that are part of the process *ClaimProcess*, are in the ready state, and for which the current user is a potential owner.
- The ID and name of each task instance.
- The description of each task instance in the default language.
- The ID and the name of the process instance of each task.

To do so, you need to join the predefined query tables TASK_DESC and PROCESS_INSTANCE as attached query tables, add the required attributes to the query table, and set the conditions accordingly.

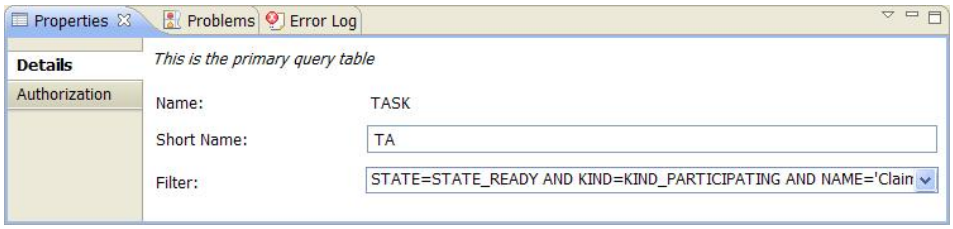
2.3.1. Specifying the primary query table filter

The query table should contain only those task instances that are in the ready state. To specify this condition, follow these steps.

1. In the editor, select the primary query table **TA** and go to the **Properties** view.

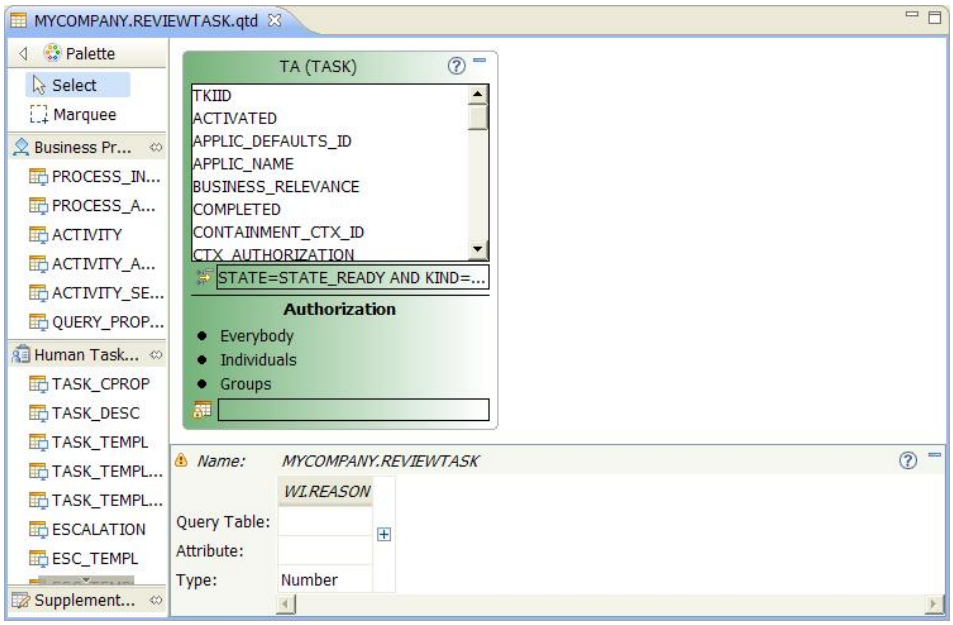


2. In the **Filter** field, enter STATE=STATE_READY AND KIND=KIND_PARTICIPATING AND NAME='ClaimProcess$ReviewHumanTask'.



Note: The filter `NAME='ClaimProcess$ReviewHumanTask'` specifies the task instances that will be part of this query table by defining the task name. The syntax for the task name for this kind of filter is `<processName>$<taskName>`.

3. In the toolbar, click **Save** (💾).



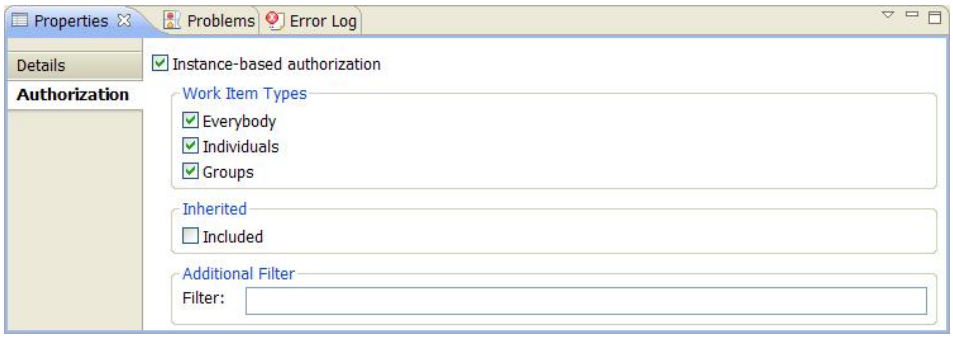
Notice that a validation warning ⚠ is shown in the editor. This is because the query table does not contain any attributes. These will be added in the following sections.

2.3.2. Specifying the authorization filter

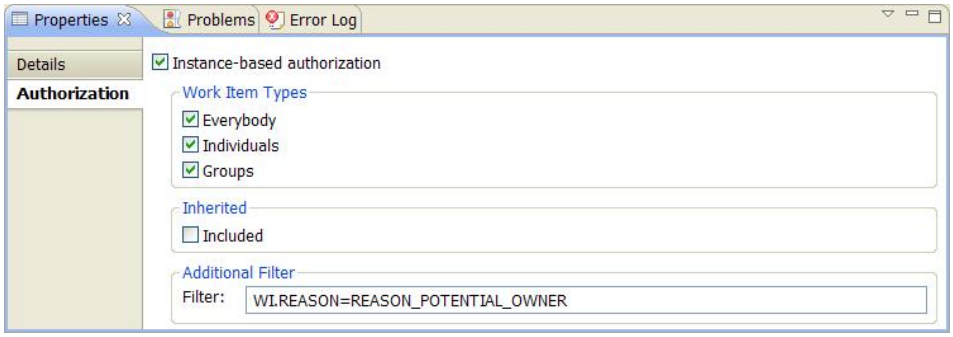
In addition to the primary query table filter, you need to specify an authorization filter to include only those tasks in the query table for which the user is a potential owner. To specify an authorization filter, complete the following steps.

1. In the editor, select the primary query table **TA,** and go to the **Properties** view.

2. Select the **Authorization** tab. The authorization settings are shown:



3. In the **Filter** field, enter `WI.REASON=REASON_POTENTIAL_OWNER`.



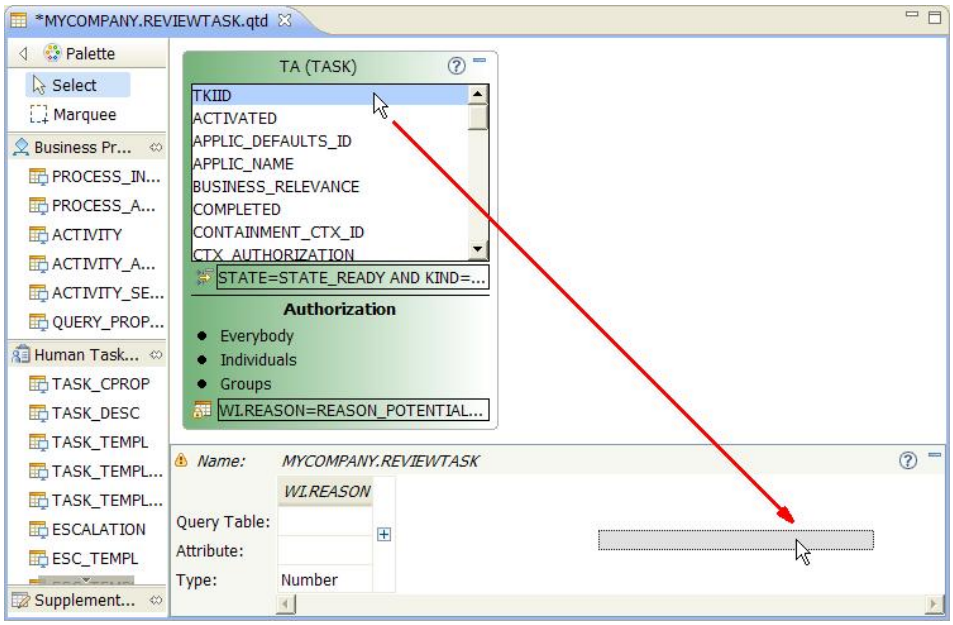
4. On the toolbar, click **Save** (💾).

2.3.3. Adding task attributes

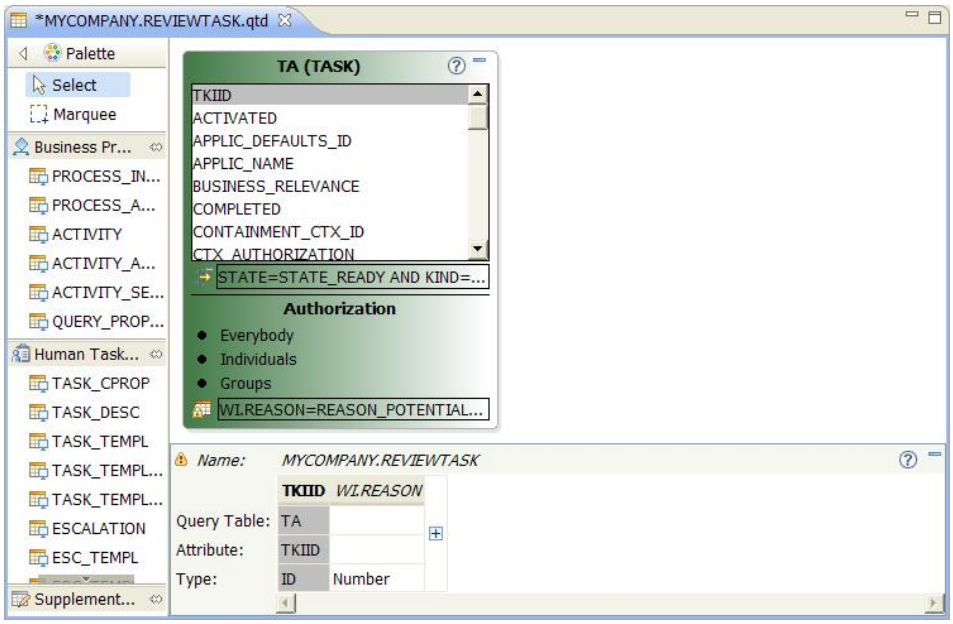
The query table should contain the ID and the name of each task, which are attributes of the primary query table TA. You can add an attribute to the query table by using drag-and-drop, or by double-clicking the attribute.

Perform the following steps to add the required attributes to the query table:

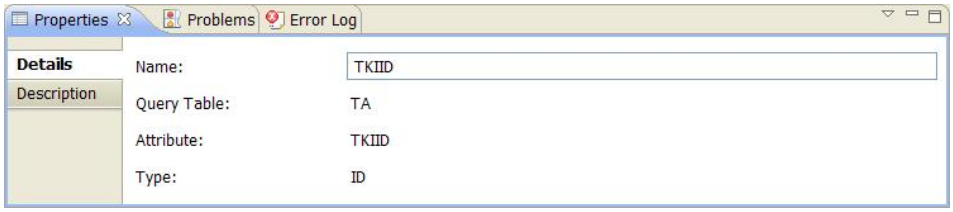
1. Select the attribute **TKIID** in the primary query table **TA**, and drop it anywhere in the table at the bottom of the editor.



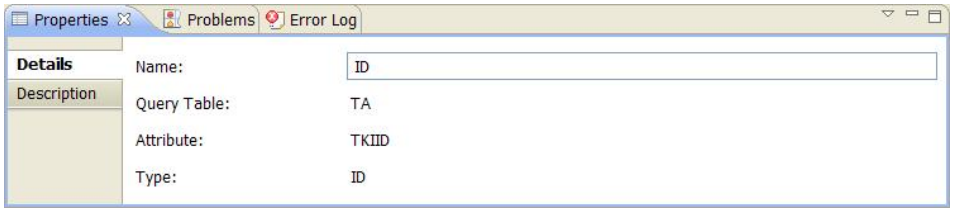
The attribute is added to the query table:



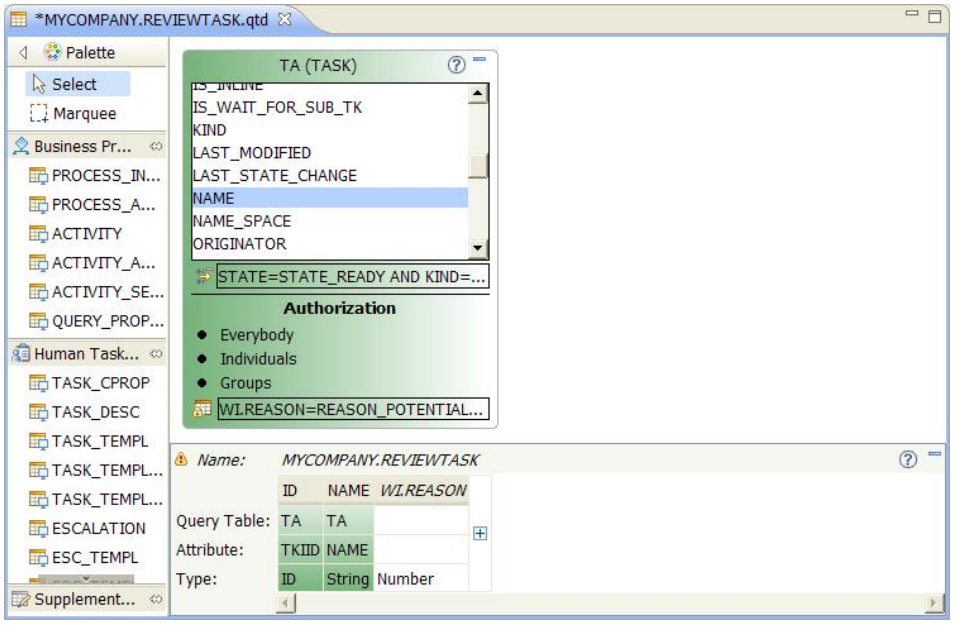
2. Go to the **Properties** view to edit the properties of the new attribute:



3. Change the **Name** of the new query table attribute to `ID`.



4. To add the task name attribute, scroll down the list of attributes in the primary query table **TA** to the **NAME** attribute, and then double-click it. The attribute is added to the query table, next to the **ID** attribute:



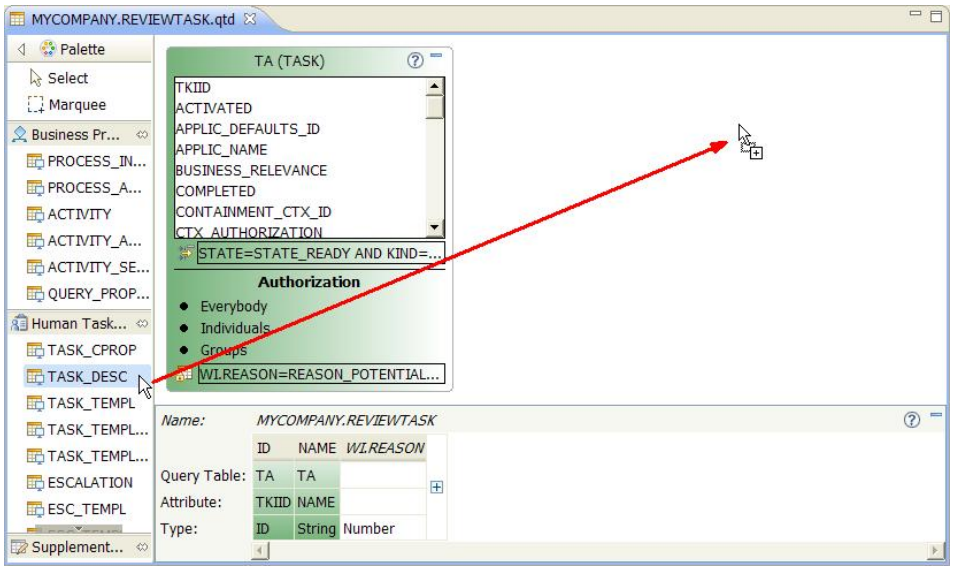
5. On the toolbar, click **Save** (💾).

2.3.4. Adding the task description

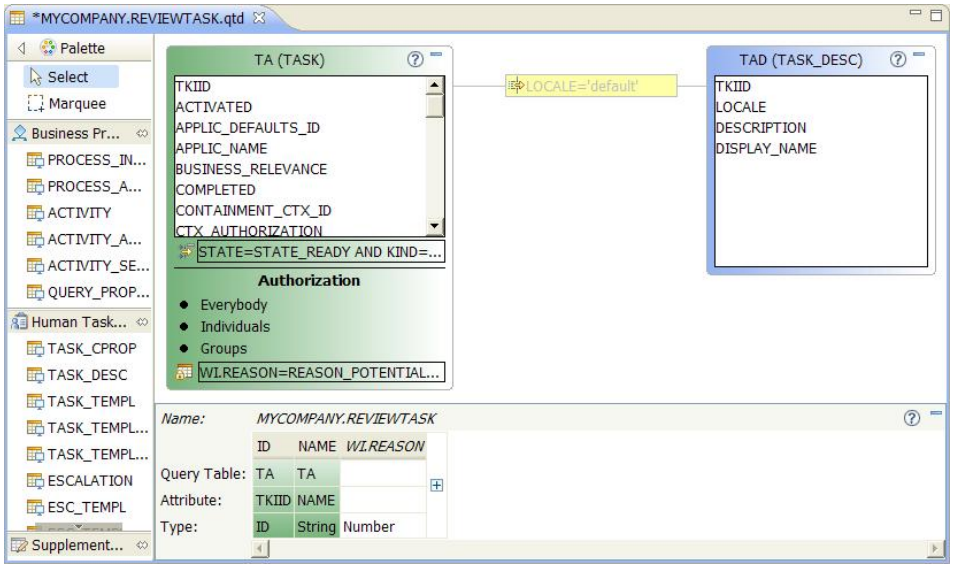
In addition to the information from the primary query table TA, the description of each task instance should also be included. To add the task description, complete the following steps.

1. Go to the **Palette** on the left of the editor.

2. In the **Human Task** section, select the **TASK_DESC** entry, and drop it on the editor canvas.

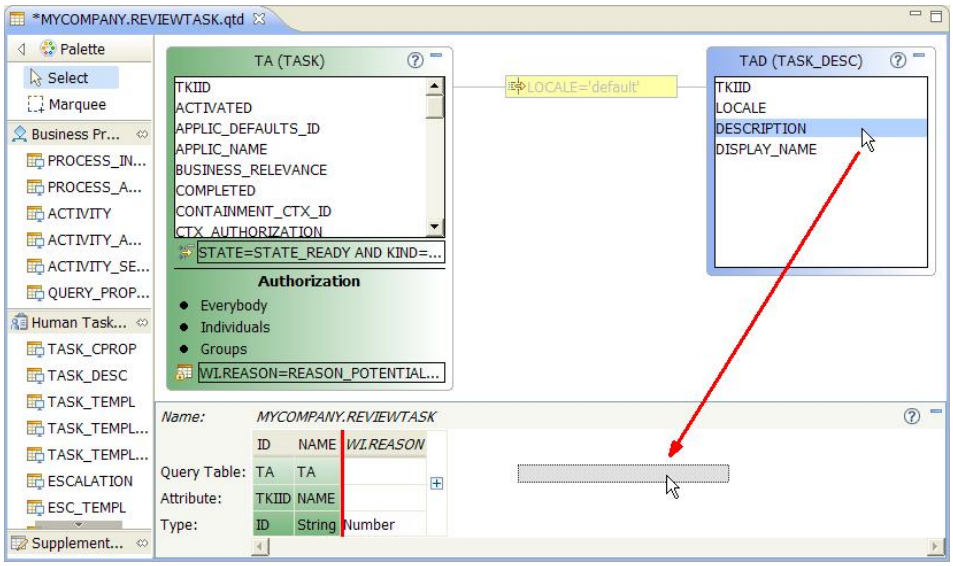


The predefined query table TASK_DESC is added as the *attached query table* TAD, which is joined to the primary query table TA. Attached query tables are blue and they are positioned by default in a column next to the primary query table.

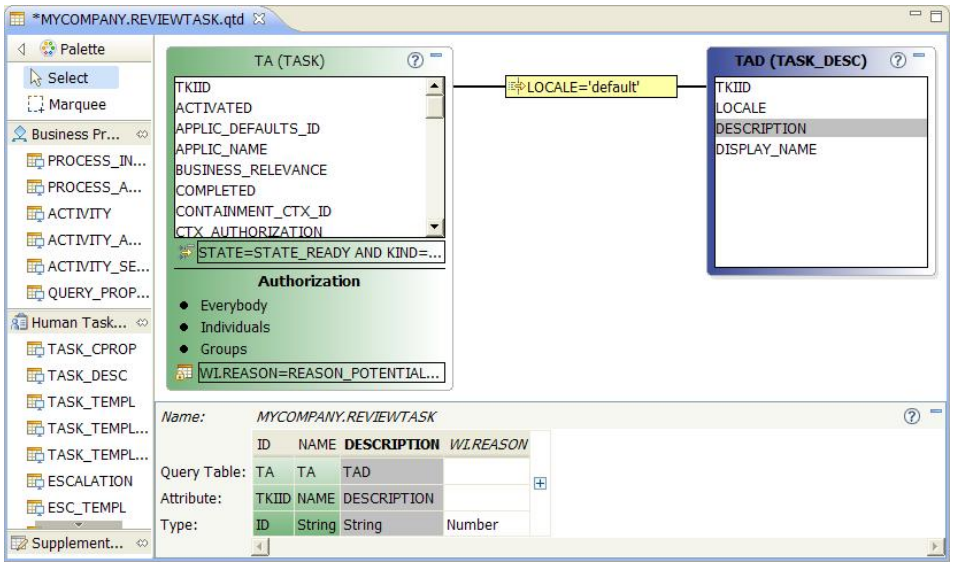


The join of the attached query table to the primary query table has predefined *selection criteria*, visualized as the yellow box on the join connection line.

3. In the attached query table **TAD**, select the **DESCRIPTION** attribute, and drop it on the table at the bottom of the editor next to the **NAME** attribute.



The position where the attribute is added to the table is marked with a red line. This is useful if you want to add an attribute in a specific position in the table.



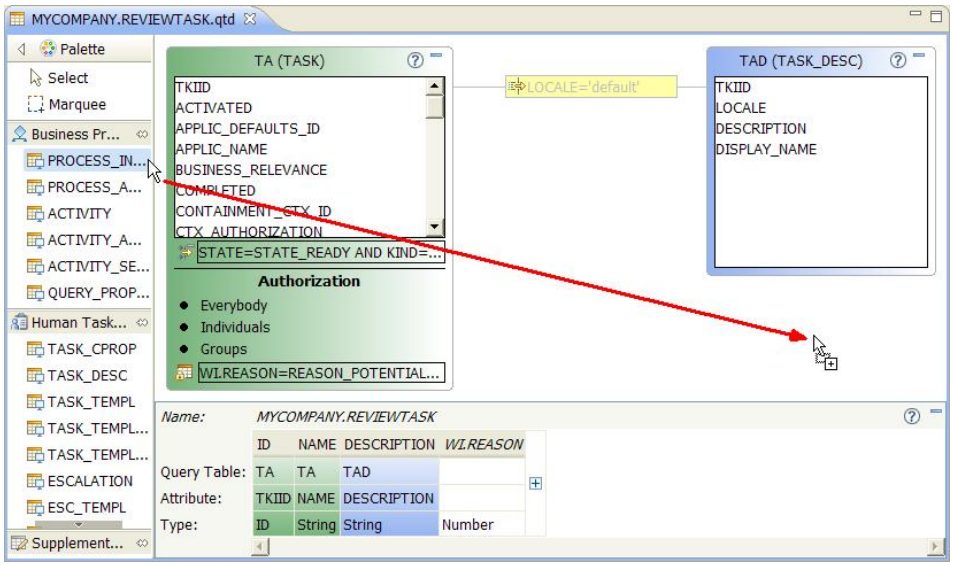
4. On the toolbar, click **Save** (💾).

2.3.5. Adding the process information

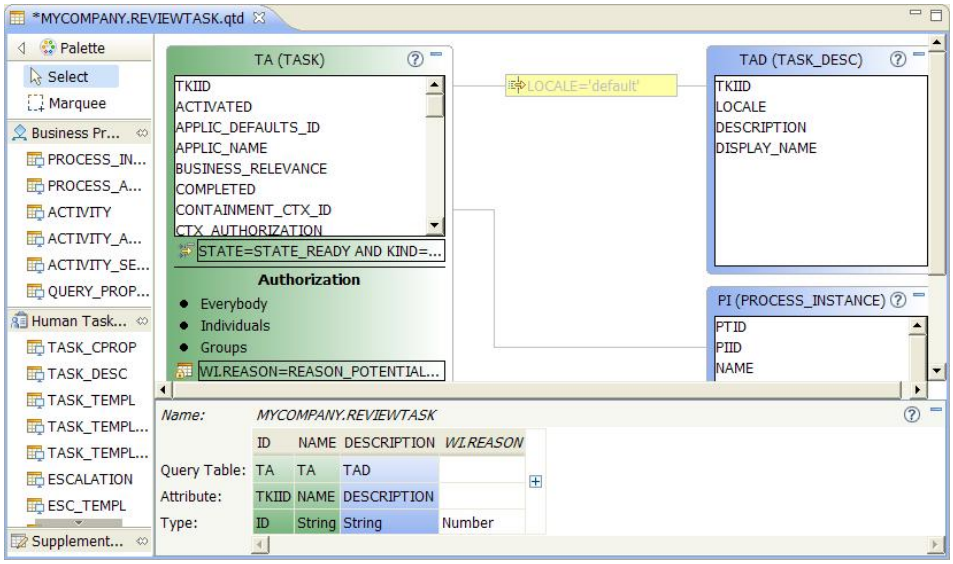
The final addition to the query table is the information about the process instance. To add the process instance ID and the name as attributes to the query table, complete the following steps.

1. Go to the **Palette** on the left of the editor.

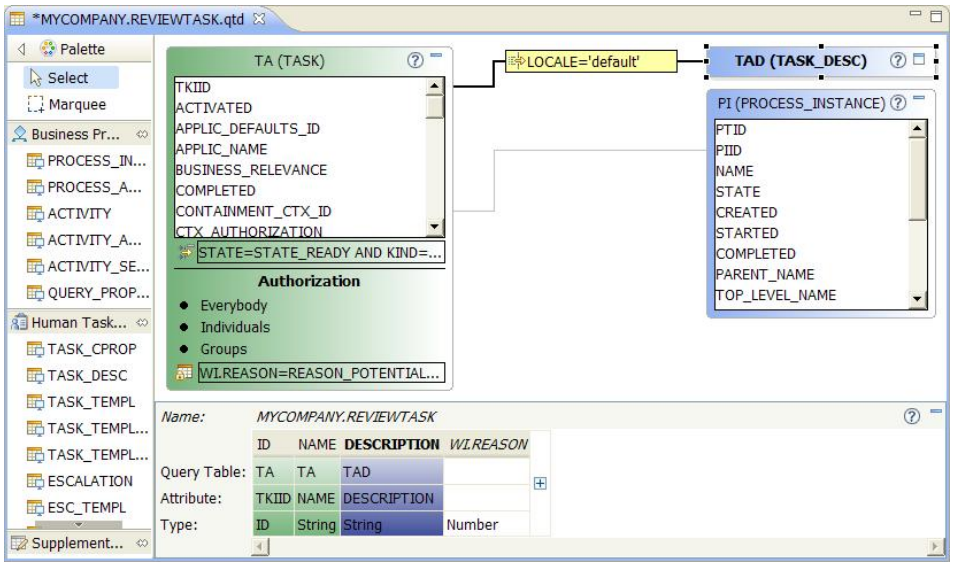
2. In the **Business Process** section, select the **PROCESS_-INSTANCE** entry, and drop it on the editor canvas.



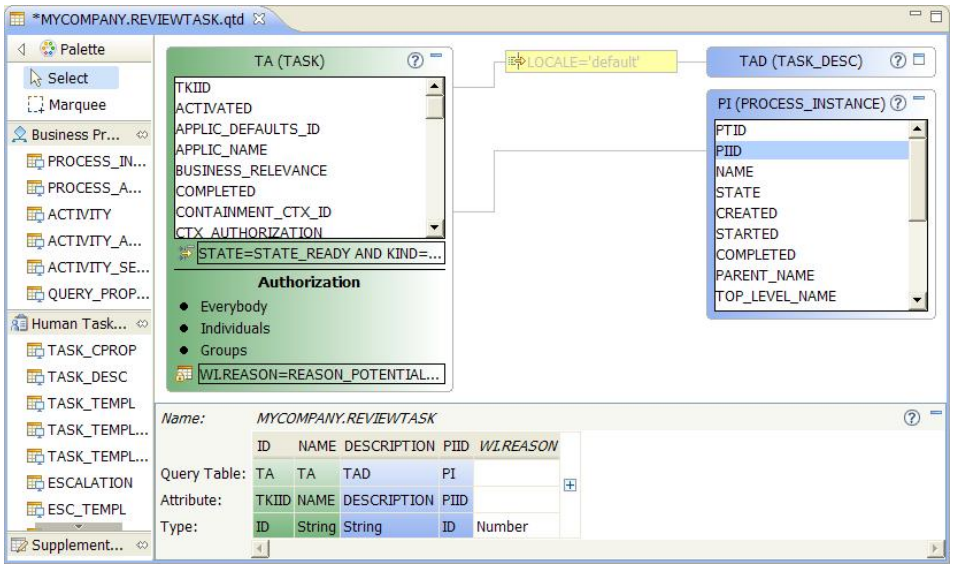
The predefined query table **PROCESS_INSTANCE** is added as the attached query table **PI**, which is joined to the primary query table **TA**:



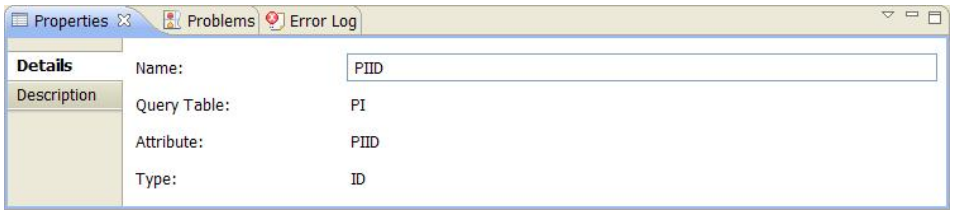
3. To get a better overview of the query table definition, you can minimize query tables. For example, you can minimize the attached query table **TAD** by clicking the minimize symbol ⊟ in the upper right corner of the query table.



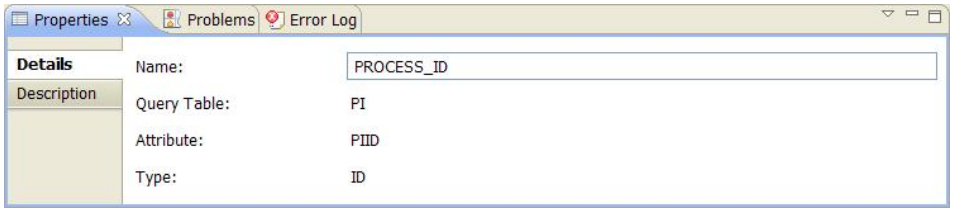
4. In the attached query table **PI**, double-click the attribute **PIID** to add it to the query table attributes table. The attribute is added next to the **DESCRIPTION** attribute.



5. Select the attribute **PIID** in the table at the bottom of the editor and go to the **Properties** view.



6. Replace the **Name** of the attribute with PROCESS_ID.



7. In the attached query table **PI**, double-click the attribute **NAME** to add it to the query table attributes table. The attribute is added next to the **PROCESS_ID** attribute.

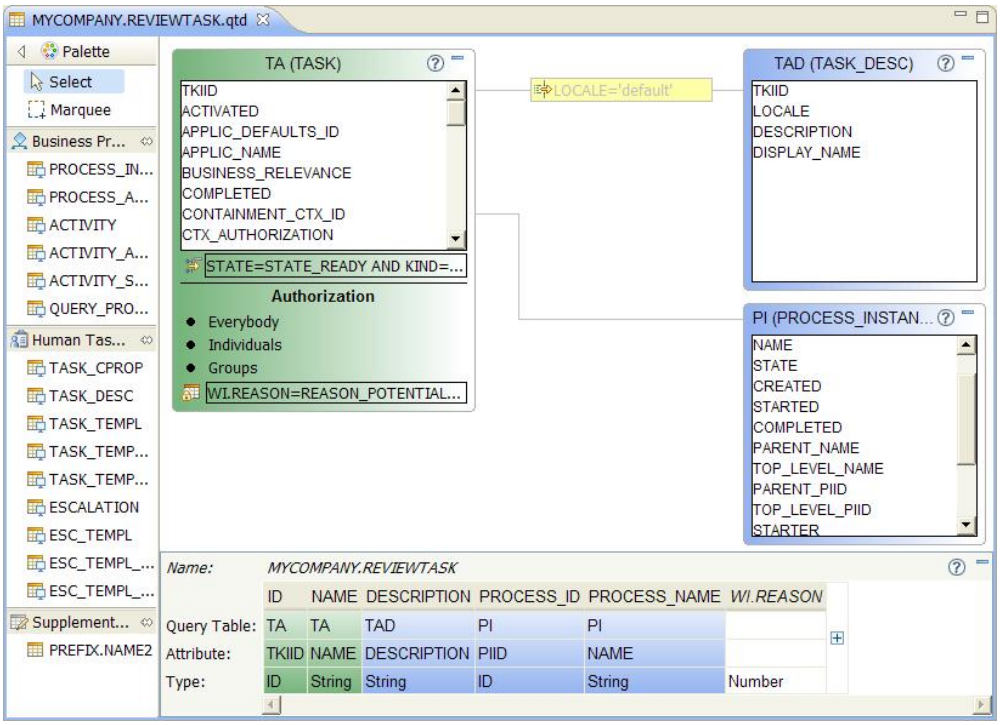
8. Select the attribute **NAME2** in the table at the bottom of the editor and go to the **Properties** view. Replace the **Name** of the attribute with PROCESS_NAME.

9. On the toolbar, click **Save** (💾).

2.3.6. Completed query table

You have created a query table that contains the following information:

- All task instances of the template *ReviewHumanTask* that are part of the process *ClaimProcess*, are in the ready state, and for which the current user is a potential owner.
- The ID and name of each task instance.
- The description of each task instance in the default language.
- The ID and the name of the process instance of each task.



2.4. Deploying and testing the query table

The Query Table Builder provides a test client for deploying and testing a query table definition file on a local Business Process Manager 7.5 server.

These instructions assume that a local Business Process Manager 7.5 server exists and that it is already running

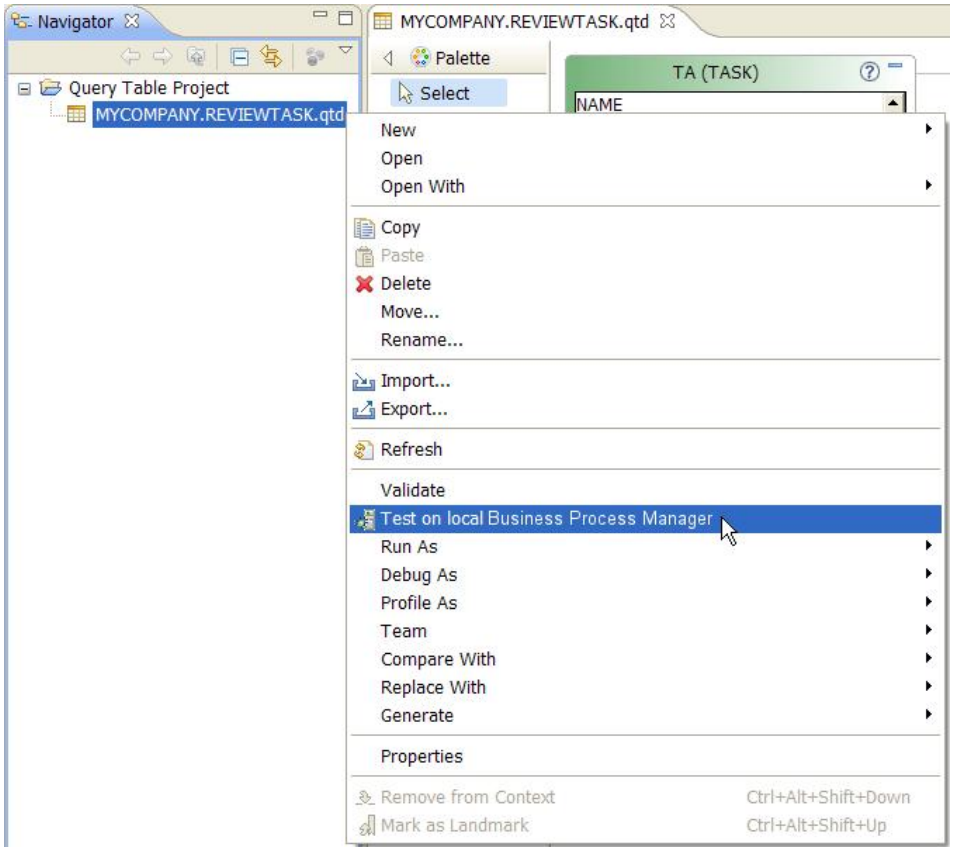
Note: Make sure that Business Process Manager contains data for the filters and conditions of the query table definition, otherwise the query client cannot display any sample output.

2.4.1. Deploying the query table

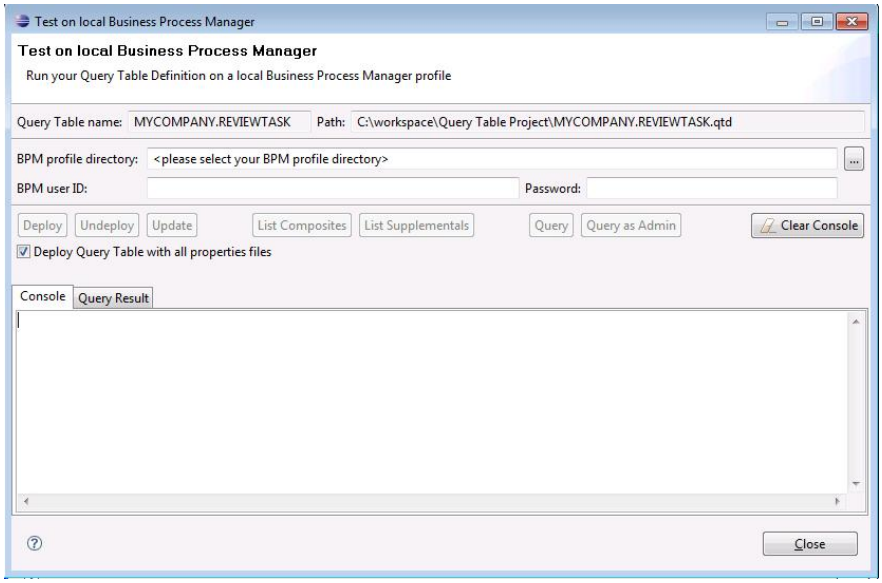
To deploy the query table definition file to the local Business Process Manager, complete the following steps.

1. Switch to the **Navigator** and right-click the MYCOMPANY.REVIEWTASK.qtd file.

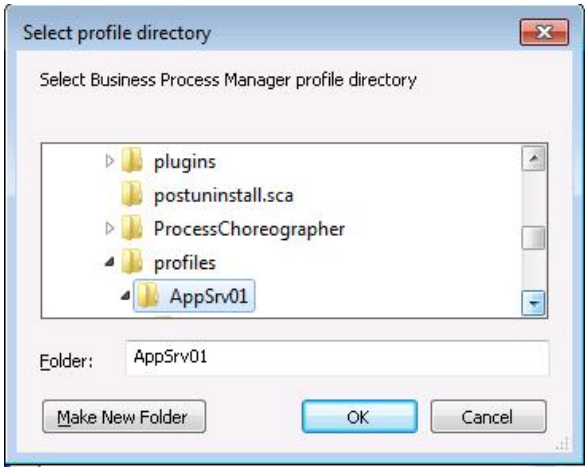
2. From the context menu, select **Test on local Business Process Manager**.



The **Test on local Business Process Manager** window opens:



3. Specify the profile directory of the local Business Process Manager by clicking **Browse** (...) next to the **BPM profile directory** field. The **Select profile directory** window opens.

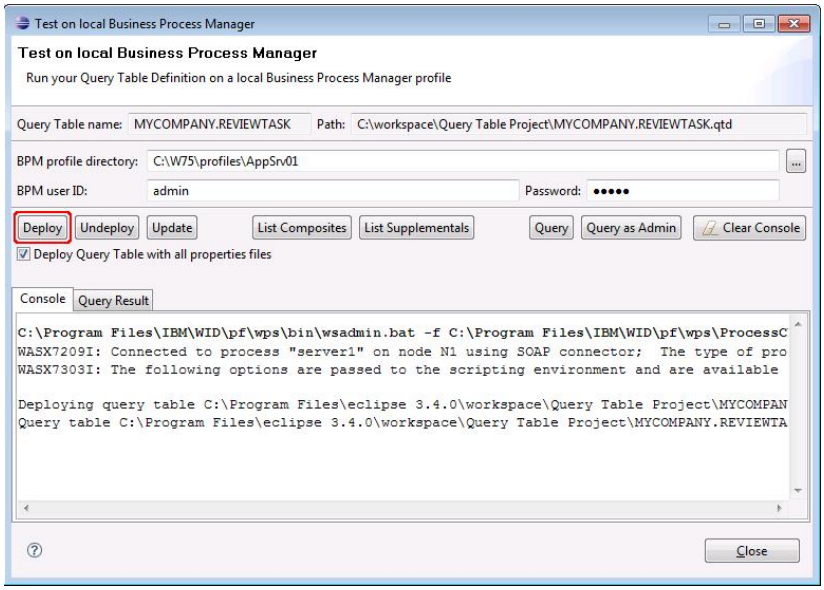


4. Select the profile directory and click **OK**. The window closes.

5. In the **BPM user ID** field, specify the user ID that is used to access the server.

6. In the **Password** field, specify the password for the user ID.

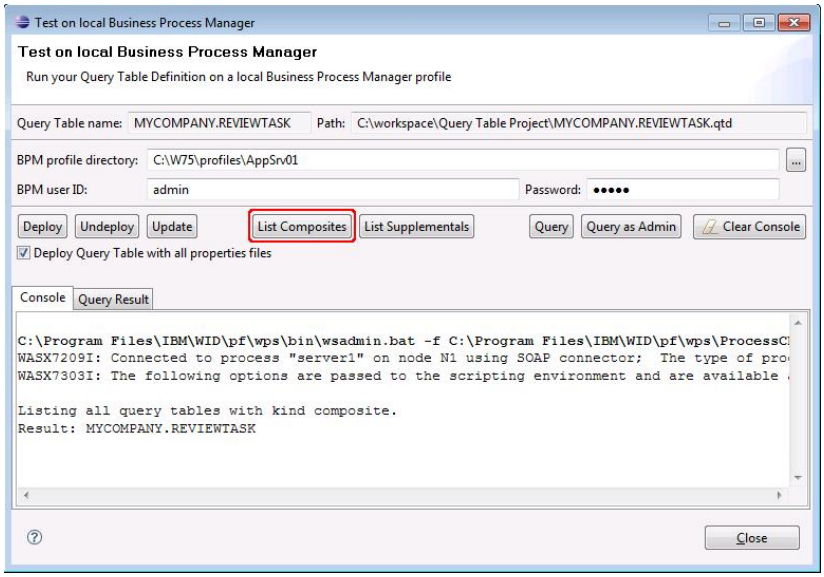
7. Click **Deploy**. The query table is deployed. The console shows the progress of the deployment, and any exceptions that occur.



Note: The console also shows the syntax of the command that is used to deploy the query table.

8. Clear the console by clicking **Clear Console**.

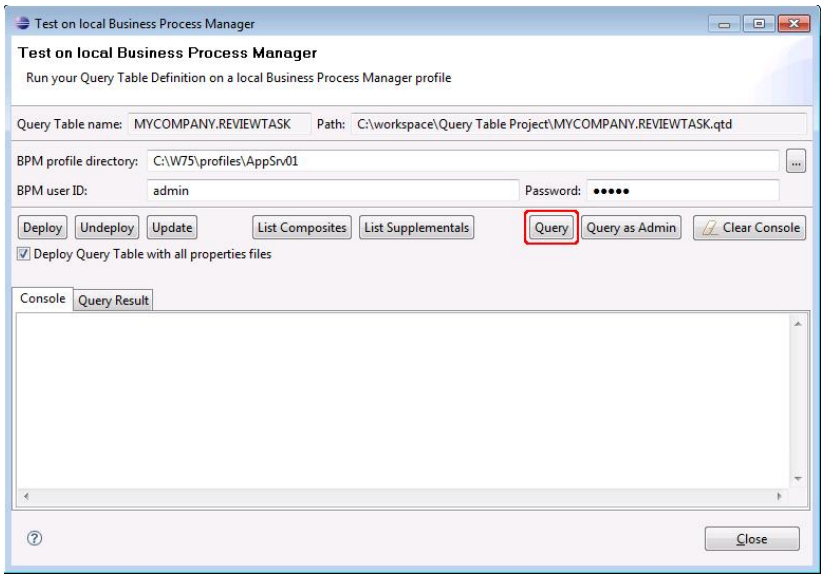
9. Click **List Composites** to see a list of all deployed composite query tables.



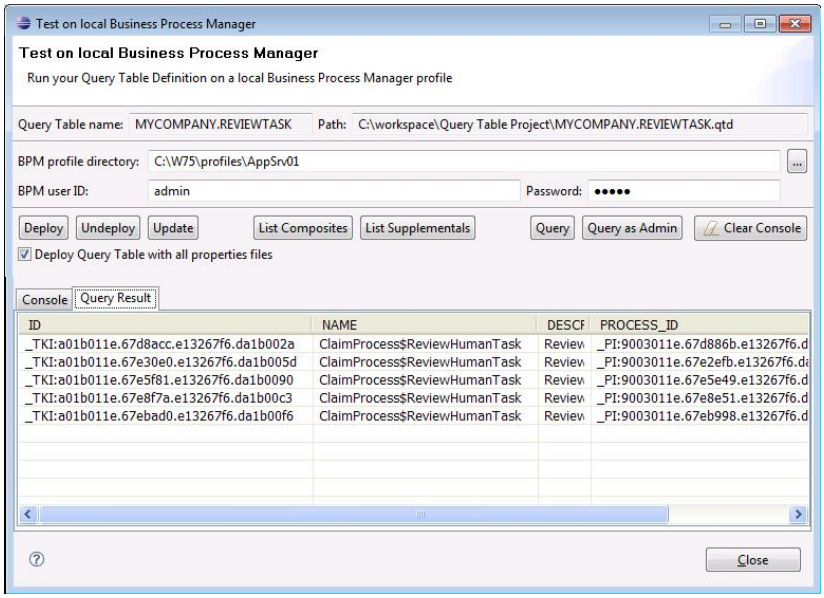
2.4.2. Testing the query table

After the query table is deployed, it can be tested. This step assumes that the Business Process Manager has one or more human tasks in the ready state. To test the query table, complete the following steps.

1. In the **Test on local Business Process Manager** window, click **Query**.



A query is created and sent to the server. Notice that the console shows the syntax of the command that is being used and the progress information. After the query is finished, the window automatically switches to the **Query Result** tab, and the result of the query is shown in a table:

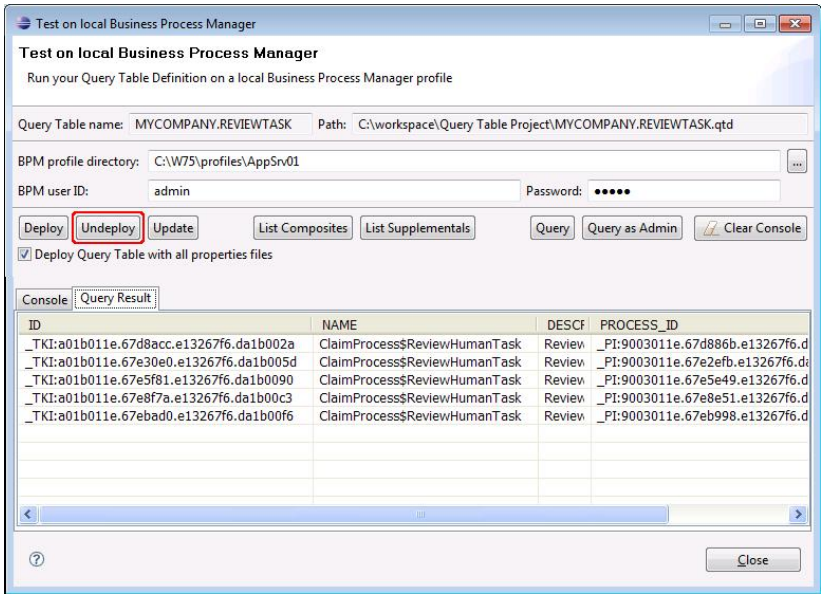


In this example, the query returns five tasks.

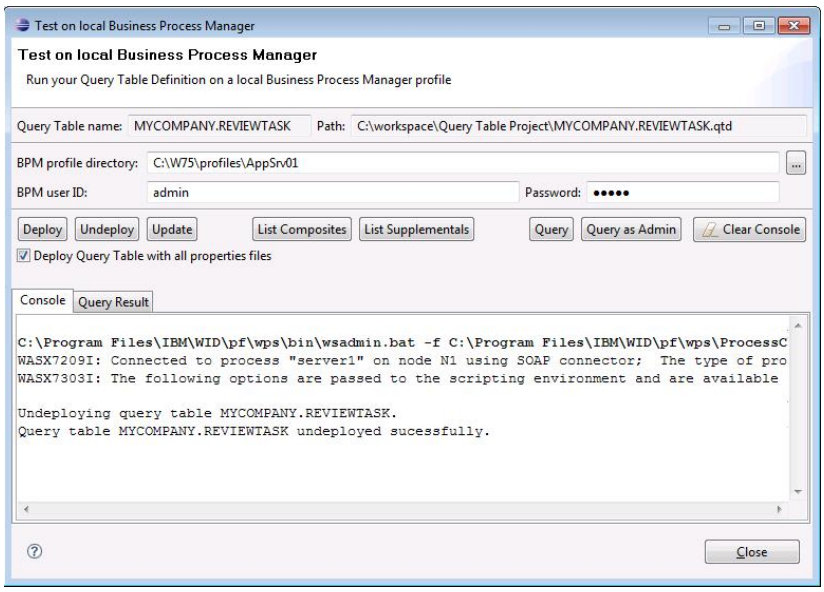
2.4.3. Undeploying the query table

Finally, you need to undeploy the query table by completing the following steps.

1. In the **Test on local Business Process Manager** window, click **Undeploy**.



The query table is undeployed. The console shows the syntax of the command that is being used and the progress information.



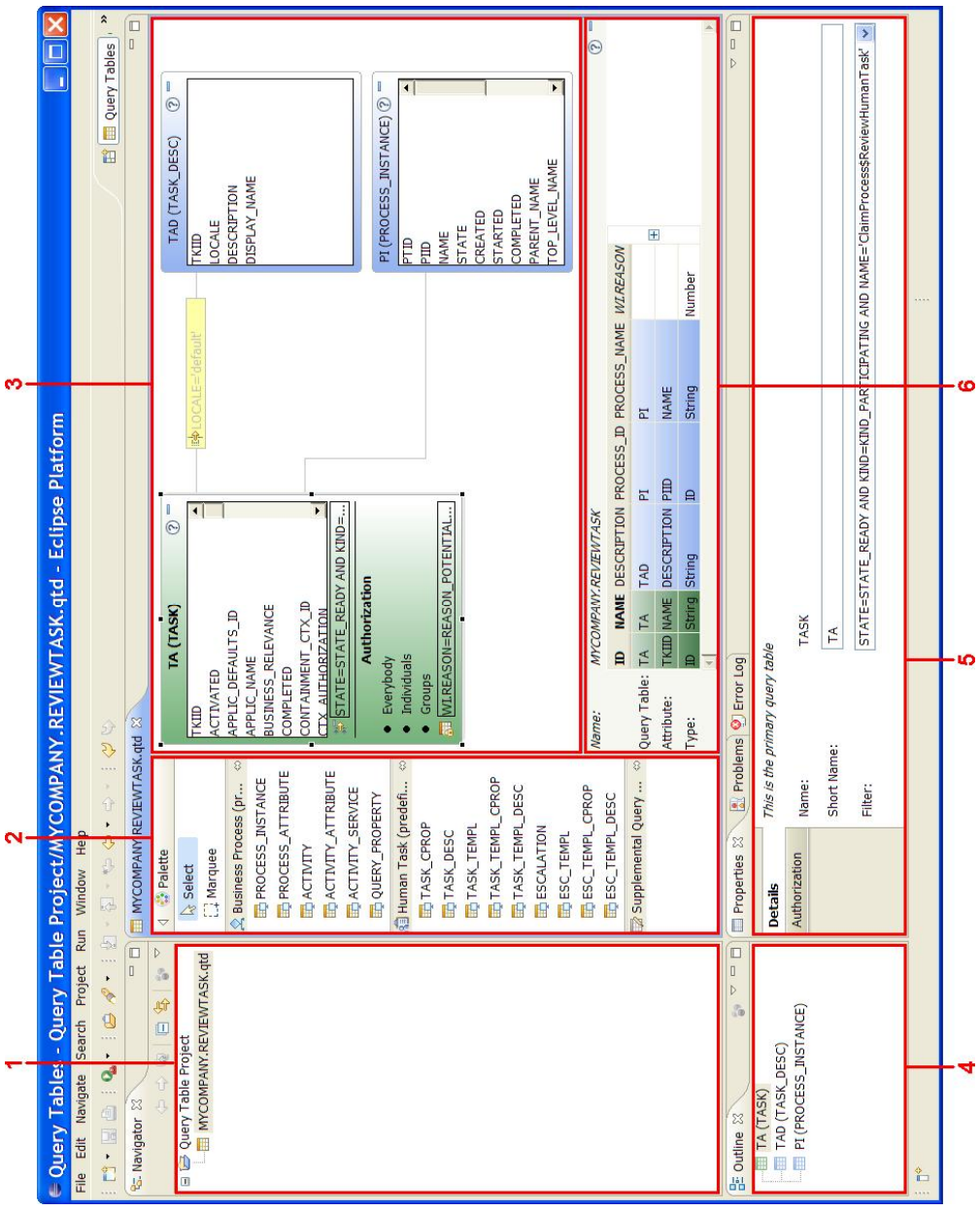
2. Close the **Test on local Business Process Manager** window by clicking **Close**.

3. Using the Query Table Builder

3.1. Overview of the Query Table Builder

This section gives you an overview of the main features of the Query Table Builder.

The following figure shows the main areas of the Query Table Builder:



The Query Table Builder has the following main areas:

1. **Navigator** – The Navigator view shows the projects and the query table definition files in those projects.
2. **Palette** – The palette contains the tools that are available for editing the query table definition, and the predefined and supplemental query tables that can be attached to the primary query table. The palette includes the following tools:
   - Select: The default cursor.
   - Marquee: Allows the selection of multiple elements.
3. **Canvas** – Predefined and supplemental query tables can be dropped on the canvas as attached query tables so that they can be included in the query table definition. Depending on the layout, the primary and the attached query tables are either aligned automatically on the canvas, or they can be repositioned as required using the freeform layout.
4. **Outline** – The Outline view lists the primary query table (with the icon) and all attached query tables (with the icon) that are part of the Query table definition.
5. **Properties view** – The view that is used to display and edit the properties of the different parts of the query table.
6. **Attributes Table** – The attributes table contains all of the attributes that are defined for the query table.

3.1.1. Query Tables perspective

The Query Table Builder offers an Eclipse perspective, which simplifies working with query tables. Whenever a new query table is created, it is checked whether the **Query Tables** perspective is being used. If this is not the case, the Query Table Builder asks you whether you would like to switch to this perspective. Although it is recommended to use this perspective while working with the Query Table Builder, it is not mandatory to do so.

Note: All examples shown in this document use the Query Tables perspective.

You can switch to the Query Tables perspective manually at any time:

- In the Eclipse toolbar, select **Window > Open Perspective > Other**.
- In the window that opens, select **Query Tables** and click **OK**.

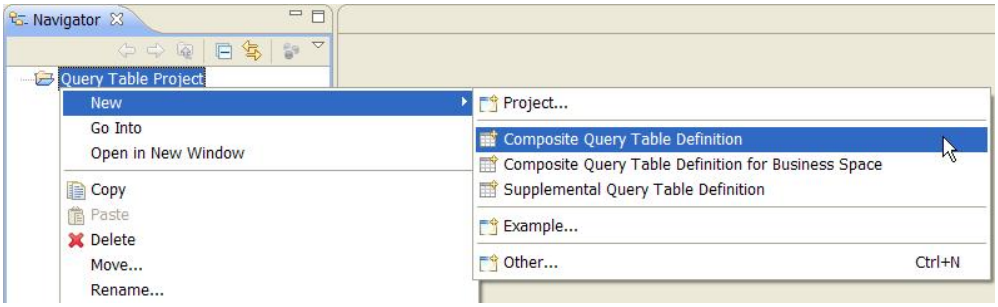
The Query Tables perspective creates a default layout for the workbench, which can be seen in the screenshot on page 28, and filters certain information that is usually not needed while creating

and editing query tables. These filters affect the **Navigator** view, as this view will only display the query table definition files (.qtd) themselves and none of the other files that are created and managed along with the query table definition. All files with the following file extensions will be hidden in the Navigator view:

- **.qtdex**: Files with this file extension are created by the Query Table Builder to store layout information.

- **.properties**: These files are created when defining display names and descriptions in different languages for localization purposes. Refer to *Localization* on page 32 for details.

In addition to these capabilities, the Query Tables perspective also offers shortcuts for creating new query table definitions in the context menu of the Navigator:



3.1.2. Help information

The Query Table Builder provides help information for most of the predefined query tables that can be used while creating a query table. Click ⓘ in the query table to navigate to the documentation of the predefined query table in the online Business Process Manager Information Center. You can also access the help information from the attributes table.

3.1.3. Minimize and maximize query tables

You can minimize the primary and the attached query tables, and the attributes table by clicking the ▬ symbol in the upper right corner of the table. To maximize them again, click the ▭ symbol.

Alternatively, the primary and the attached query tables can also be minimized and maximized by double-clicking their title.

3.1.4. Automatic Layout

By default, the Query Table Builder uses an automatic layout, in which the primary and the attached query tables are aligned in two columns: the first column contains only the primary query table, and the second column contains the attached query tables.

In this layout, you cannot reposition the primary query table, but you can resize it. If you resize the primary query table, the

attached query tables are automatically repositioned to preserve their position relative to the primary query table.

The attached query tables are aligned in one column next to the primary query table. When a new query table is attached, it is automatically added to the bottom of this list. You can change the order of the query tables in this list by selecting a query table and dropping it on a new position. You can also resize each of the attached query tables.

3.1.5. Freeform Layout

The Query Table Builder also provides a freeform layout. To switch between the layout types, right-click on the canvas and select **Switch to Freeform Layout** or **Switch to Automatic Layout.**

If the layout style is switched, the primary query table and all attached query tables are reset to a default location and size. The default arrangement for the freeform layout is a grid.

In the freeform layout, you can reposition and resize the primary and attached query tables as needed. These settings are saved when you save the query table definition.

3.1.6. Query table validation

The Query Table Builder validates the current query table whenever the query table definition file is saved. If you try to deploy a query table definition file with one or more validation errors, it will result in errors during deployment. However, a query table that contains validation warnings can still be deployed.

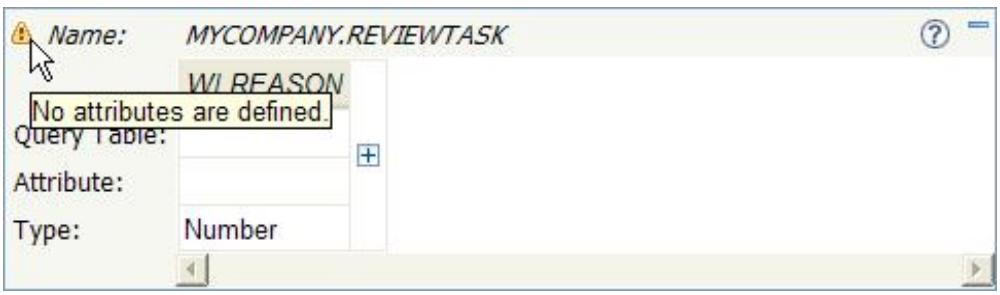
All of the warnings and errors for a query table definition file can be found in the standard Eclipse **Problems** view. To display this view, select **Window > Show View > Problems**.

The Query Table Builder also displays validation markers in a query table to indicate where the problems are. You can get more information about the problem by hovering with the cursor over these markers. The validation markers look like this:

- - Warning marker
- - Error marker

If an invalid value for a specific property caused the validation error, validation markers are also shown in the **Properties** view.

Whenever a validation error or warning concerns the entire query table and not just a specific part of it, the validation markers are displayed in the upper left corner of the attributes table. The following figure shows an example:



In this example, the warning is given because the query table does not have any attributes apart from the implicitly available WORK_ITEM attributes, such as WI.REASON (see *Changing the properties of the primary query table* on page 41).

To revalidate the query table after changes have been made, save the query table definition file again.

3.1.7. Properties

The properties of the different parts of a query table will always be shown and edited in the standard Eclipse **Properties** view. The contents of this view depends on the currently selected element in the editor.

To show and edit the properties of an element, open the **Properties** view in one of the following ways:

- Select the element by clicking on it. In the Eclipse toolbar, select **Window > Show View > Other**. In the window that opens, select **General > Properties**.
- Right-click on the element and select **Show in Properties**.

3.1.8. Localization

It may be required to provide the *localization* of a query table, i.e. to offer a *display name* and a *description* in different *languages* for the query table itself and for each of its attributes.

In the Query Table Builder, the **Properties** view of the query table itself and of each attribute that is defined in the query table allows to create and manage display names and descriptions in different languages. Refer to *Changing the properties of the query table* on page 36 and to *Changing the properties of an attribute* on page 48, respectively.

A language that is being used in this context is specified by its name, which has to be defined according to the following format:

<language>_<country>_<variant>

This format allows the definition of different specializations of a language, as *country* and *variant* are optional. Some examples:

Language	Description
en	English
en_AU	English, Australia
en_GB	English, United Kingdom
de_AT	German, Austria
de_DE	German, Germany
de_DE_EURO	German, Germany, Euro

For each language, a property file (file extension .properties) is created. The display name and description values that are defined for the query table and the attributes for this specific language are stored in this file. Upon deployment of the query table on the Business Process Manager, it is possible to deploy it along with all language property files that exist. Refer to *Deploying query table definitions* on page 63 for details. If the **Query Tables** perspective is being used, the property files will be hidden and are not shown in the **Navigator**, as described in *Query Tables perspective* on page 29.

By convention, a *default language* is defined for each query table. This language does not have a name that defines which language it is. This file contains default values for display names and descriptions if the values are not defined in a specific language.

3.2. Working with composite query tables

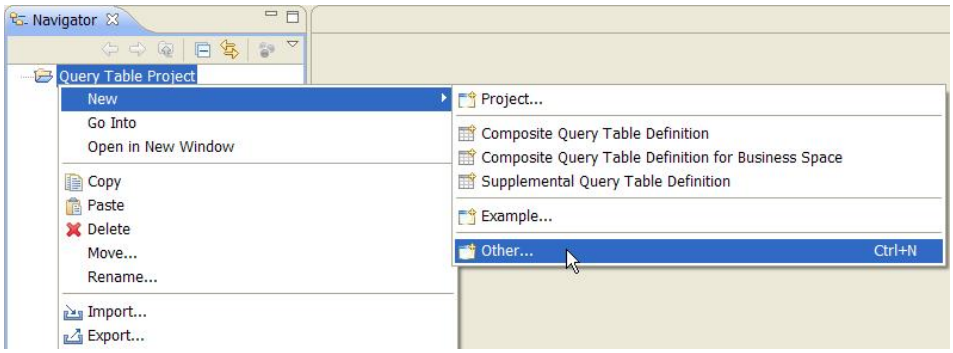
This section describes how to create and edit composite query tables in the Query Table Builder.

3.2.1. Creating a composite query table

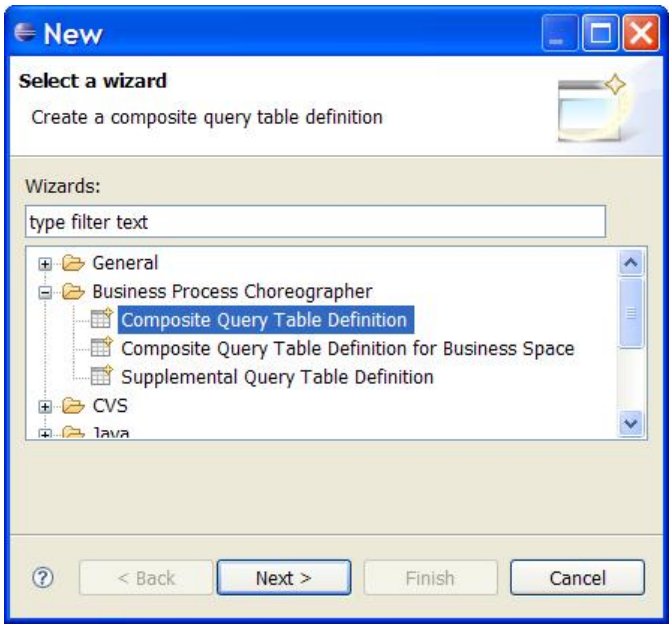
You can create a query table definition (.qtd) file in any type of project, for example, a simple project, or a Java project. However, you can put a query table definition only in the root directory of a project, not in any subdirectories or packages.

To create a composite query table, complete the following steps.

1. Switch to the **Navigator**, right-click the project in which the query table definition file is to be created, and select **New > Other**.

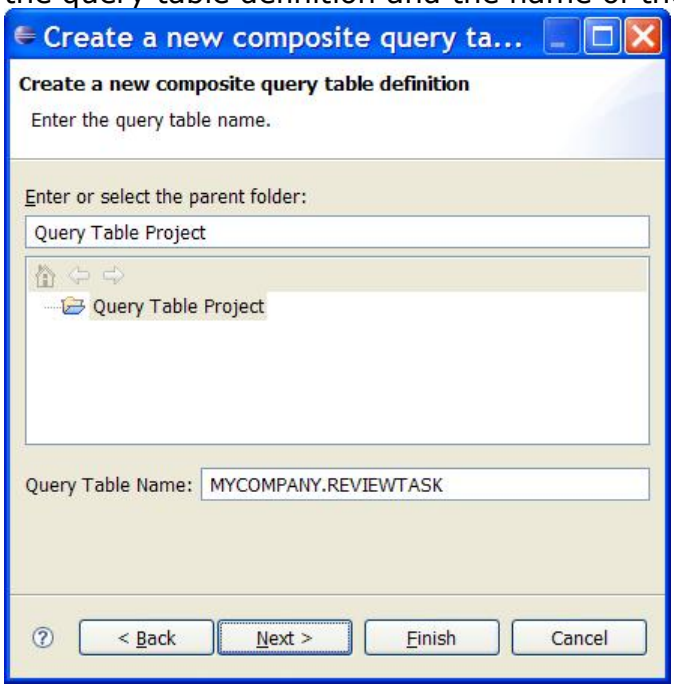


2. In the window that opens, specify the type of query table that is to be created:



In the **Business Process Choreographer** section, select **Composite Query Table Definition**, and click **Next**.

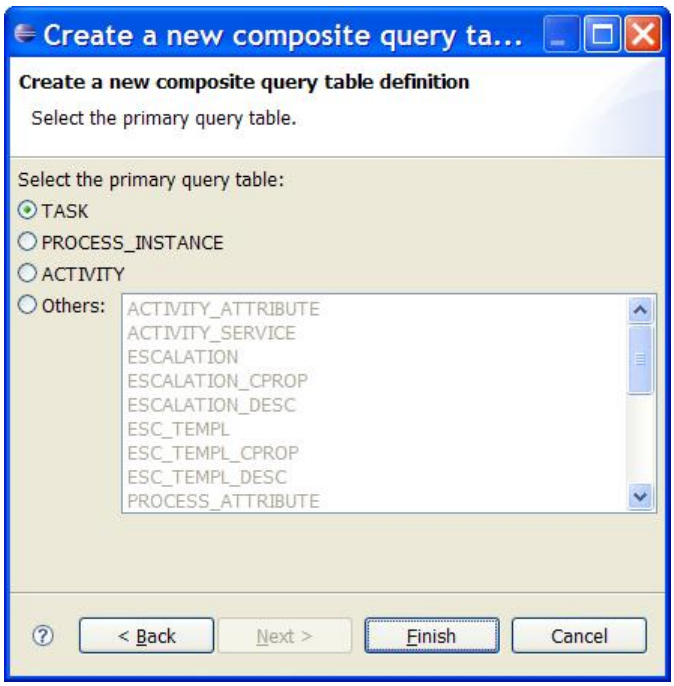
3. Select the parent folder – the root directory of a project – for the query table definition and the name of the query table:



In the **Query Table Name** field, enter the name of the new query table. The name must be in uppercase and have the format *PREFIX.NAME*. It can have a maximum of 28 characters. The prefix and the name can contain numbers, but they cannot start with one. You should not change the name of the query table after the query table definition file is created.

4. Click **Next**.

5. Select the primary query table:



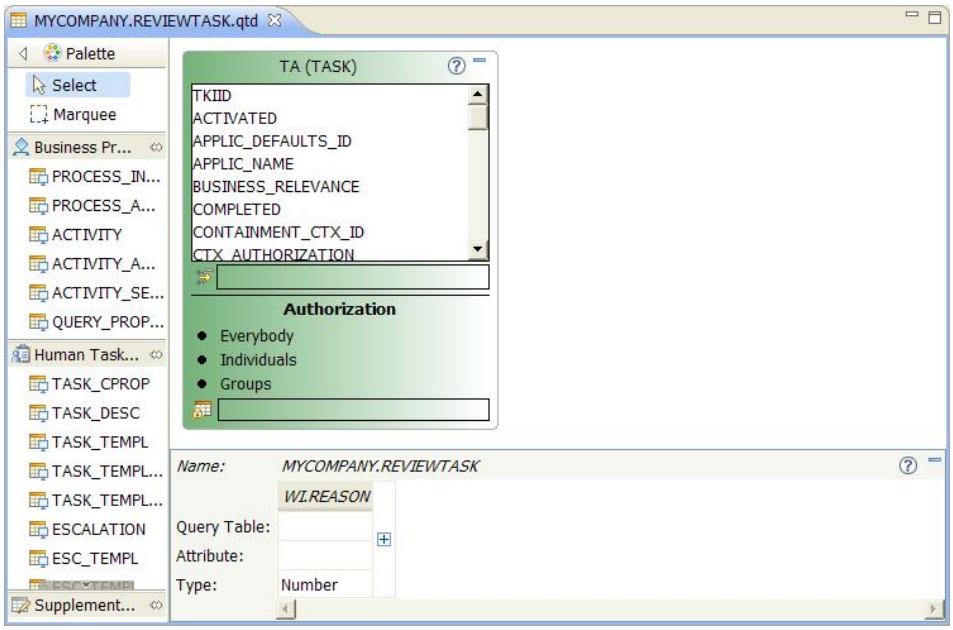
Select one of the commonly-used primary query tables **TASK**, **PROCESS_INSTANCE**, **ACTIVITY**, or select **Others** and choose from the list of available predefined query tables.

Note: You cannot change the primary query table after the query table is created.

6. Click **Finish**.

   The composite query table is created, and the editor opens the new query table definition file. In addition to the query table definition file, a file with the extension .qtdex is also created. This file stores the layout information for the query table, such as the position and size of the primary and attached query tables.

   The initial view of the new query table looks similar to the following figure:

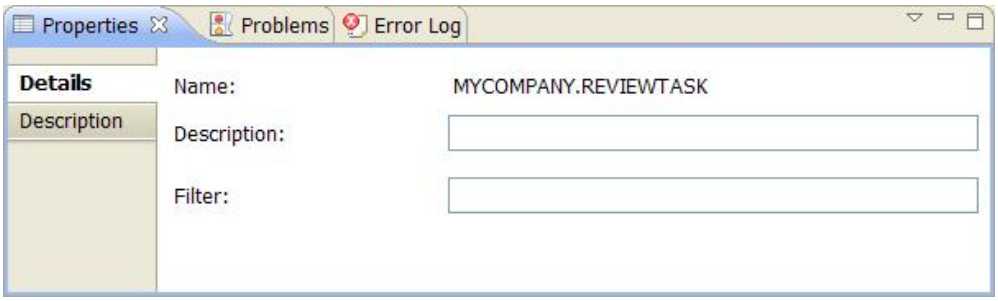


The primary query table is already placed on the canvas.

3.2.2. Changing the properties of the query table

To show and edit the properties of the entire query table, select either the canvas or the attributes table and go to the **Properties** view.

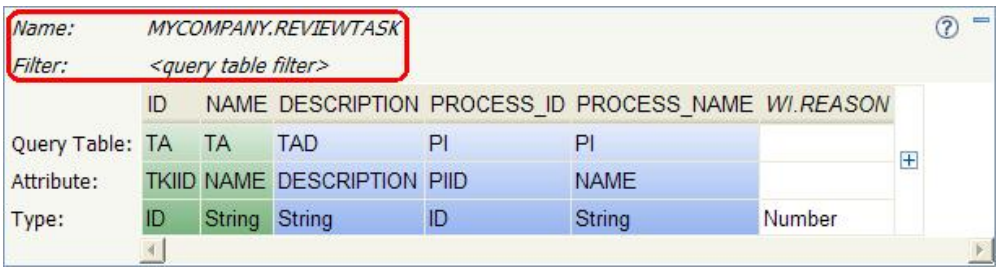
The Properties view of the query table consists of the **Details** and the **Description** tabs. The **Details** tab looks like this:



The following properties are available in the **Details** tab:

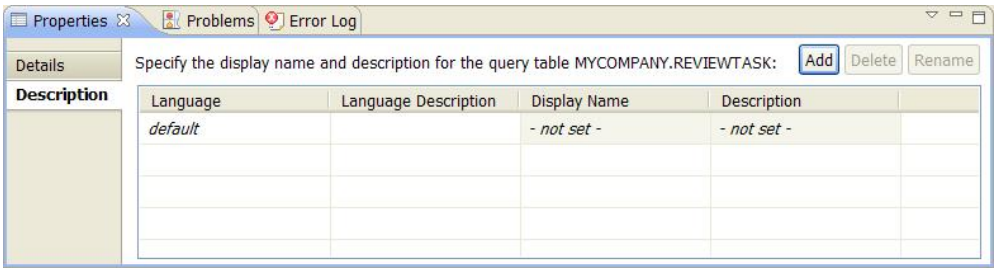
Property Name	Editable	Description
Name	No	The name of the query table. This is the name that was specified when the query table was created. You cannot change the name after the query table is created.
Description	Yes	A description of the query table.
Filter	Yes	The filter for the query table. Any of the attributes that are defined in the attributes table can be included in the filter. Refer to the links specified on page 3 for details on the syntax and performance impact of using this filter.

The name and the filter of the query table that can be viewed and edited in this Properties view are also displayed at the top of the attribute table:



If a filter is not defined, then the **Filter** row in the attributes table is hidden.

The **Description** tab looks like this:



This view lists the languages (refer to *Localization* on page 32 for details) that have been defined for this query table, as well as for each language the display name and the description for this query table. Initially, this view contains only the **default** language.

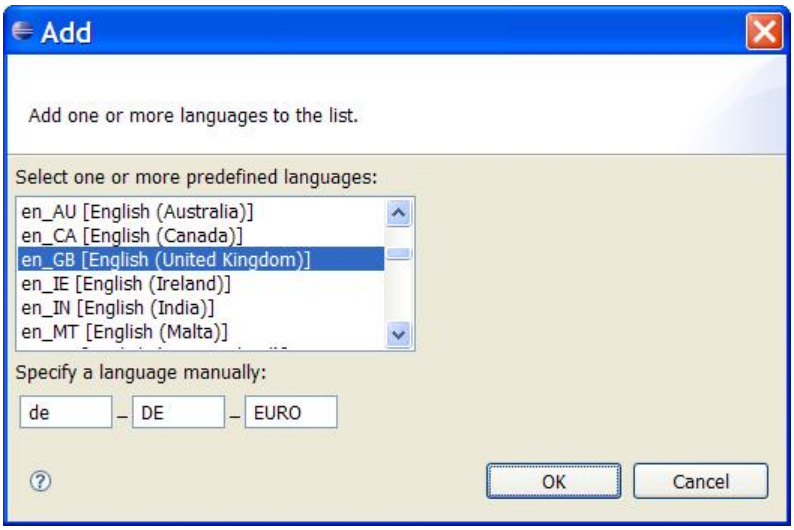
Refer to the next section on how to work with languages.

3.2.3. Creating, modifying and deleting a language

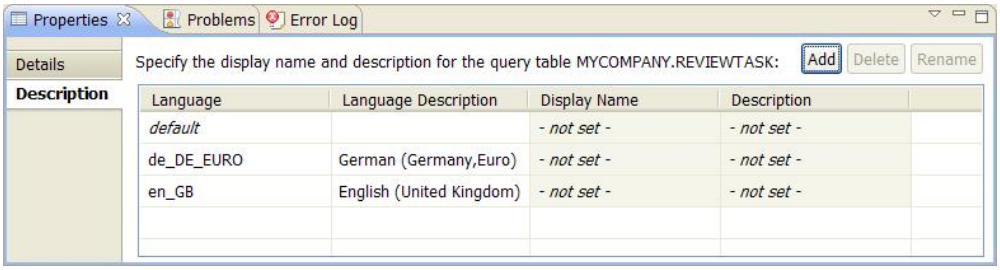
The languages table in the **Description** tab of the query table's **Properties** view contains the following information:

Column	Description
Language	The name of the language.
Language Description	A description of the language. This description is created automatically by matching predefined values for the language, country and variant parts of the language to a human readable description. This description can not be changed.
Display Name	The display name of the query table in this language.
Description	The description of the query table in this language.

To create a language, click **Add**. The **Add** window opens.



In this window, you can select one or more languages from a list of predefined languages, and you can define a language manually by specifying the different parts of a language name (language, country, variant). To add the language(s), click **OK**. The table of available languages is updated:

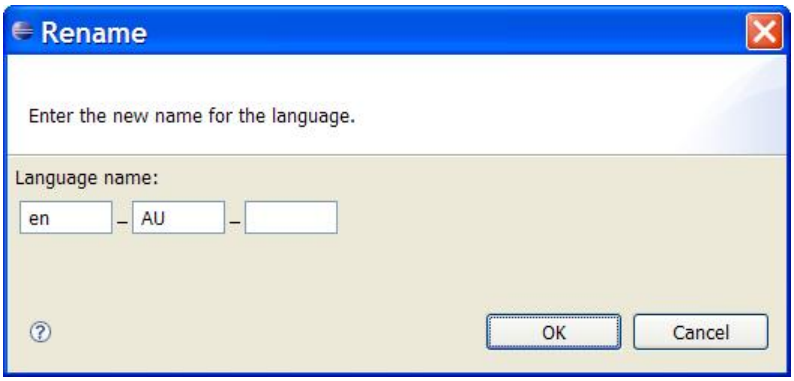


This new language is now available for the query table as well as for all attributes defined in it. So when you switch for example to one of the query table's attributes, the newly added language will also be listed in the language table of its Description tab.

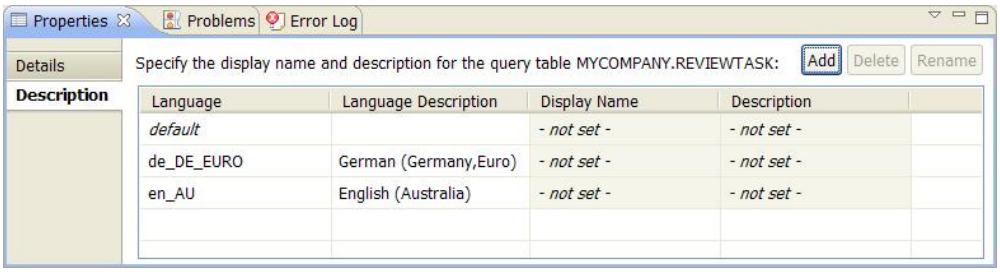
To delete a language, select the language in the table and click **Delete**.

Note: Deleting a language in the table causes this language's property file to be deleted. As a consequence, all display names or descriptions that have been specified elsewhere for this language (e.g. for an attribute) are deleted as well.

When working with languages, it may be necessary to rename a language. To do so, select the language from the table and click **Rename**. The **Rename** window opens.



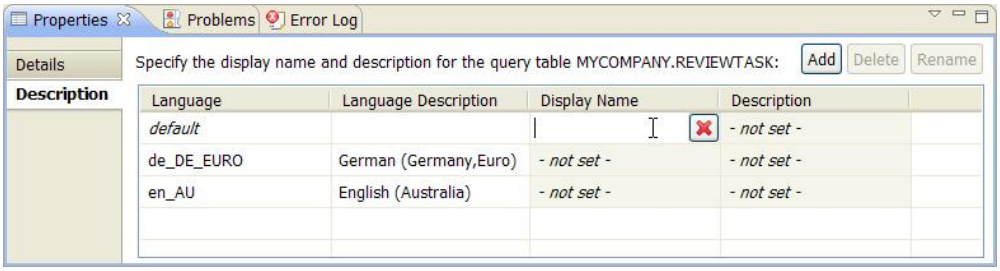
Modify the current name of the language and click **OK**. The table is updated:



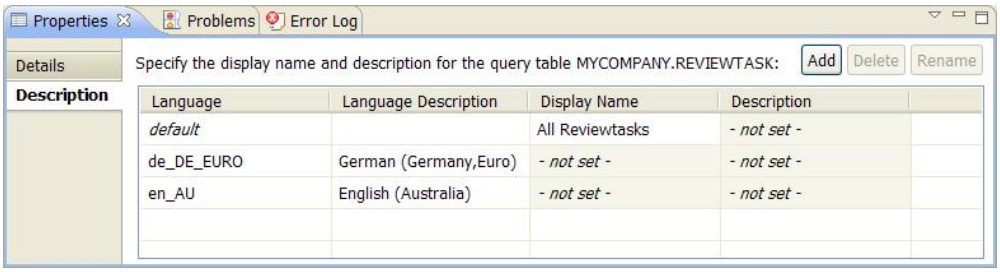
Note: Renaming a language also renames the property file for this language.

For each of the languages in the table, a **Display Name** and a **Description** can be defined. Initially, these properties are defined as "- not set -", which means that the property file does not contain an entry.

To set or modify the Display Name or Description of a language, click into its cell.



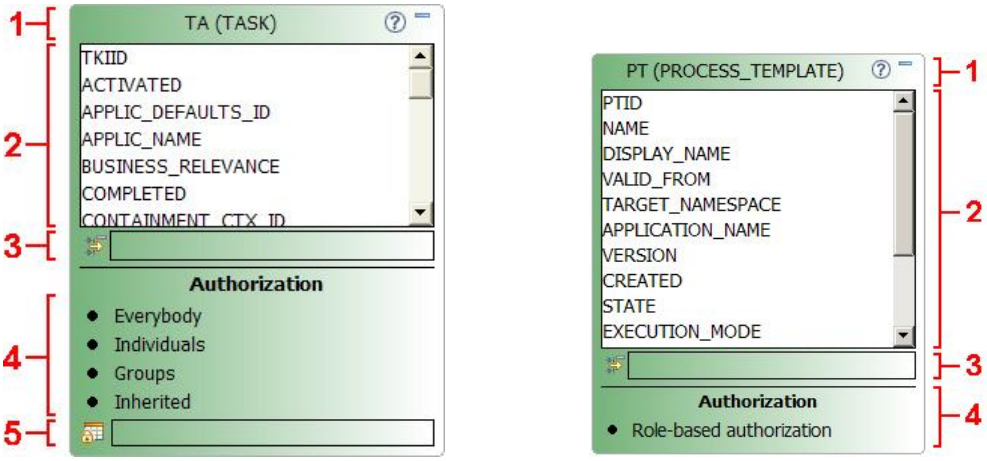
The property is activated (the property file now contains an entry) and you can specify a value. To leave this editing mode, click anywhere in the table.



To change a Display Name or Description back to "- not set -", and thus to delete the entry from the property file, click in its cell and click  .

3.2.4. Primary query table

The primary query table is represented by the following figure:



The different sections of a primary query table are defined as follows:

Section	Name	Description
1	Title	The title of the primary query table is made up of its short name and of its name in parentheses: **short name (name)**.
2	Attributes List	The list contains all of the attributes that are defined in the primary query table. Attributes from this list can be added to the query table attributes.
3	Primary Query Table Filter	The filter for the primary query table.
4	Authorization Types	The authorization types that are included in the query table. For primary query tables containing instance information, you may specify work item types like *everybody* and *individuals*. For primary query tables containing template information, *role-based authorization* is the only possible option.
5	Authorization Filter	The authorization filter defined for the query table. Only available for primary query tables containing instance information.

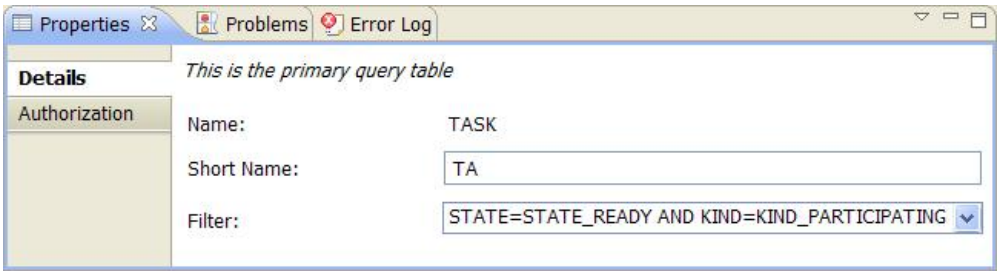
Note that sections 4 and 5 are displayed only if authorization is enabled.

You cannot delete the primary query table.

3.2.5. Changing the properties of the primary query table

To show and edit the properties of the primary query table, select it and go to the **Properties** view.

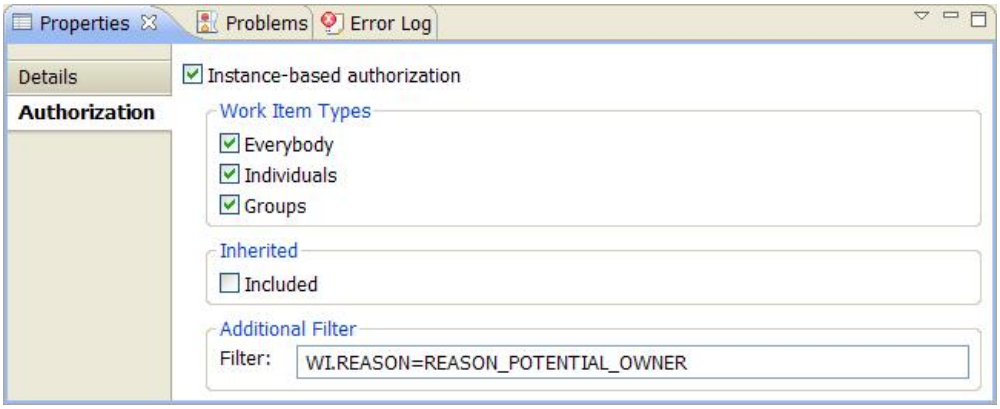
The Properties view of the primary query table consists of the **Details** and the **Authorization** tabs. The **Details** tab looks like this:



The following properties are available in the **Details** tab:

Property Name	Editable	Description
Name	No	The name of the primary query table.
Short Name	Yes	The short name for the primary query table, which is used to identify it.
Filter	Yes	The filter for the primary query table. All of the attributes in the attribute list can be used in this query.

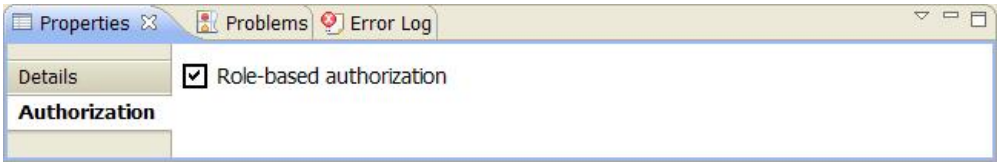
For primary query tables containing instance information, the **Authorization** tab looks like this:



The following properties are available in the **Authorization** tab:

Property Name	Editable	Description
Instance Based Authorization	Yes	This option controls whether instance-based authorization is enabled. • If this option is set, only those objects are included in the query results for which the user is authorized. • If this option is not set, all of the objects are included. Note that bypassing instance-based authorization should be used with caution! If authorization is enabled, the authorization settings are shown in the primary query table, and the attributes of the predefined query table WORK_ITEM are available in the attributes table. **Note:** All other authorization properties can be edited only if this option is set.
Everybody	(Yes)	If checked, Everybody work items are considered. Everybody work items provide authorization for all authenticated users.
Individuals	(Yes)	If checked, work items are considered that provide authorization for individual users.
Groups	(Yes)	If checked, work items are considered that provide authorization for groups (Group work items).
Inherited Included	(Yes)	If checked, process readers and process administrators may access inline human tasks and related objects.
Filter	(Yes)	The authorization filter. Any of the attributes that are defined in the predefined query table WORK_ITEM can be included in the filter. These attributes must be prefixed with 'WI.'. For example, the REASON attribute must be referenced as **WI.REASON**.

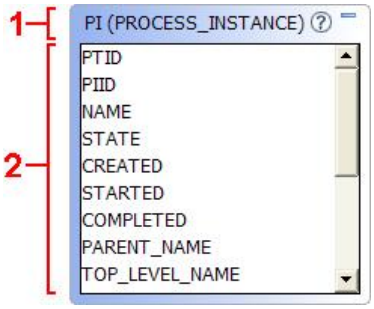
For primary query tables containing template information, the **Authorization** tab looks like this:



You can only specify whether to use role-based authorization or not. There are no further options available.

3.2.6. Attached query tables

An attached query table looks like this:



The different sections of an attached query table are defined as follows:

Section	Name	Description
1	Title	The title of an attached query table has the format: **short name (name)**.
2	Attributes List	The list contains all of the attributes that are defined for the attached query table. Attributes from this list can be added to the query table attributes.

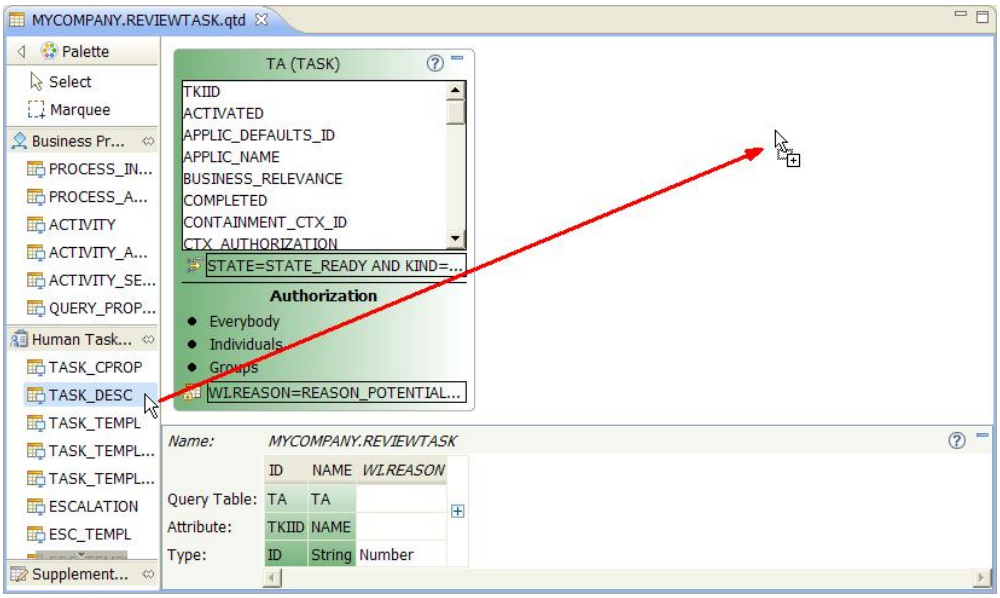
3.2.7. Adding and deleting an attached query table

You can add any of the query tables that are available in the palette to the query table as attached query tables. The palette contains:

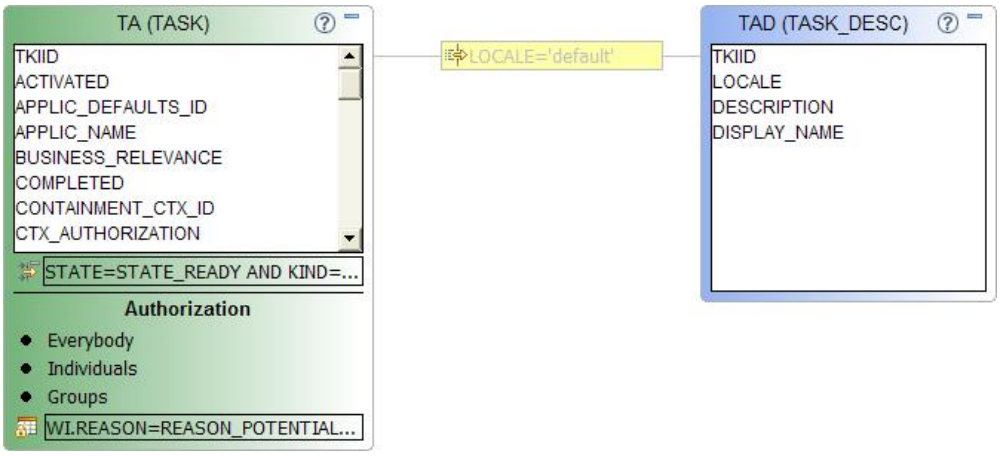
- In the **Business Process** and **Human Task** sections: All of the predefined query tables that can be attached to the primary query table.
- In the **Supplemental Query Tables** section: All of the supplemental query tables that are in the same project as the current query table definition.

The contents of the palette therefore depend on the type of the primary query table and the supplemental query tables in the project.

To attach a predefined or supplemental query table to the primary query table, select its entry on the palette and drop it on the canvas.

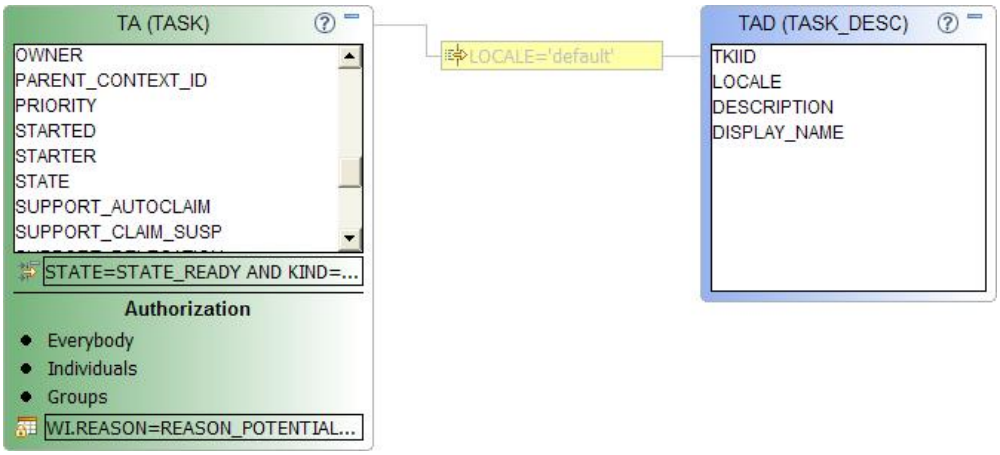


The new attached query table is joined to the primary query table by a grey line from the attached query table attribute to the primary query table attribute. The following figure shows an example of a join connection:



In this figure, the attached query table TAD is joined to the primary query table TA via the attached query table attribute TKIID and the primary query table attribute TKIID. The join also displays the attached query table selection criteria in the yellow box on the connection line. If selection criteria are not defined, the box is not shown. An exception to this rule is when it is recommended for the specific join of the primary query table and attached query table to specify selection criteria. In this case, the box contains **<Selection Criteria>**.

When one of the two attributes connected by the join connection line is not visible, for example, because one of the tables is minimized, the join connection is attached either to the top or the bottom of the table, as shown in the following figure:



A predefined or supplemental query table can be attached multiple times. In this case, a number is added to the default short name of the new attached query table.

To delete an attached query table, do one of the following:

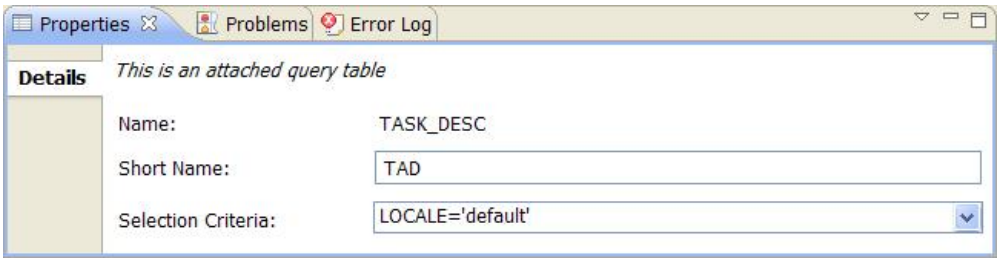
- Right-click anywhere in the attached query table (but not in the list of attributes), and select **Delete** from the context menu.
- Select the table and press the delete key (**DEL**) on your keyboard

The attached query table and the join connection are deleted.

3.2.8. Changing the properties of an attached query table

To show and edit the properties of an attached query table, select it and go to the **Properties** view.

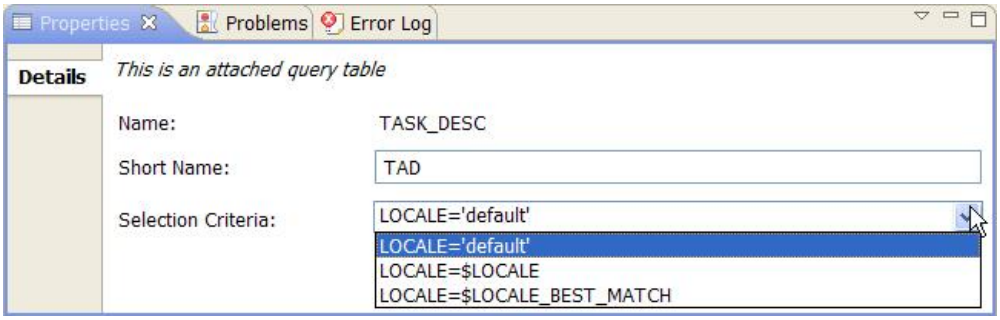
The Properties view of the attached query table consists of the **Details** tab:



The following properties are available in the Properties view:

Property Name	Editable	Description
Name	No	The name of the attached query table.
Short Name	Yes	The short name for the attached query table, which is used to identify it. This name must be unique within the composite query table.
Selection Criteria	Yes	The selection criteria of the attached query table. All of the attributes that are part of its attribute list can be used in the selection criteria.

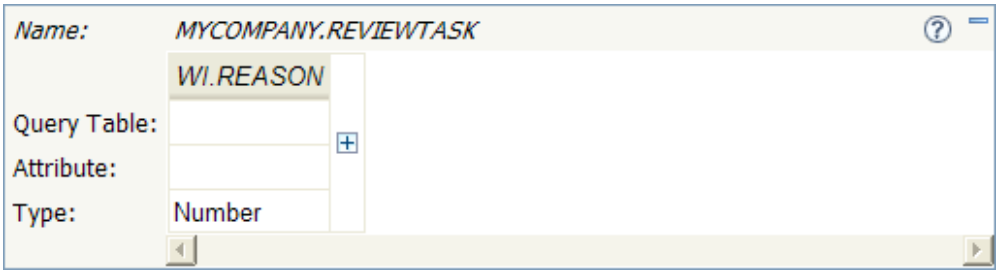
For certain attached query tables, the Query Table Builder provides predefined selection criteria. They can be viewed by expanding the drop-down box:



When a new query table is attached and a predefined selection criteria exists, it is automatically inserted. You should customize the selection criteria to address your needs.

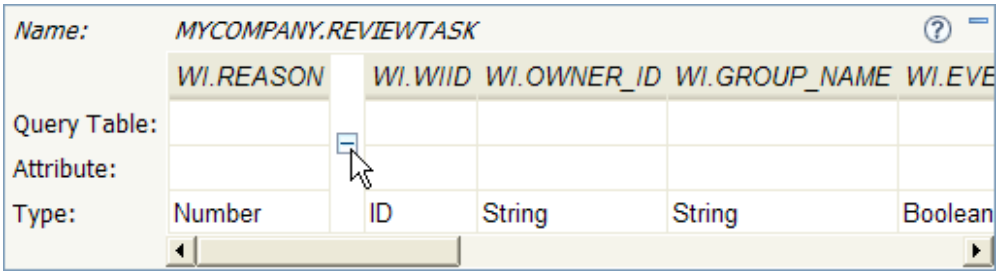
3.2.9. Working with the attributes table

The attributes table contains the attributes that are defined for the query table. When a new query table definition is created with the default settings, the attributes table looks like this:



The attributes table shows only the name of the query table (see *Changing the properties of the query table* on page 36), and the **WI.REASON** attribute. If instance-based authorization is enabled for the query table, the attributes of the WORK_ITEM predefined query table are available in the query table attributes (see *Changing the properties of the primary query table* on page 41). These attributes have the prefix "WI.". You cannot move, change, or delete these attributes.

Because the most commonly-used work item attribute is WI.REASON, all other attributes are hidden by default. To display the other work item attributes, click ⊞ next to the WI.REASON attribute.



To hide these attributes again, click ⊟.

If authorization is disabled in the properties of the primary query table, then none of the work item attributes are available.

For each attribute, the attributes table displays the short name of the query table from which the attribute was added (**Query Table**), the original name of the added attribute (**Attribute**), and the type of the attribute (**Type**). The header of each attribute in the attribute table shows the name of the attribute in the query table.

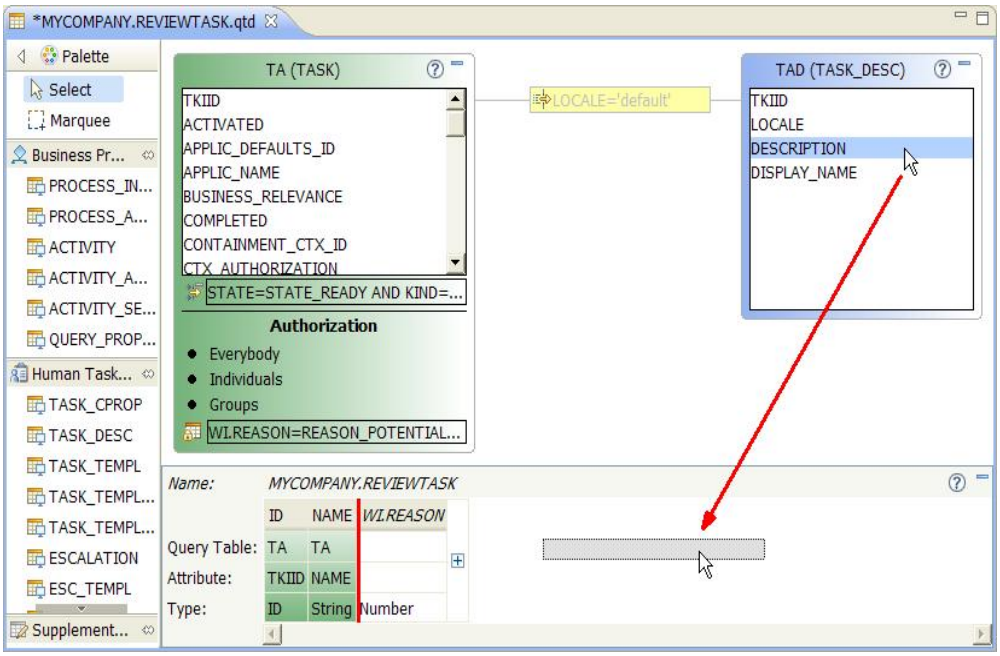
3.2.10. Adding attributes to and deleting attributes from the query table

You can add attributes from the primary and the attached query tables to the query table list of attributes. To add an attribute, complete the following steps.

1. Select the attribute in the list of attributes of the primary or the attached query table.

2. Add this attribute to the attributes table either by double-clicking on the attribute, or by using drag-and-drop.

If you double-click the attribute, it is added to the end of the list of attributes. If you use drag-and-drop, you can determine the position of the attribute within the attributes table:



In the example, the **DESCRIPTION** attribute from the attached query table **TAD** is added to the attribute table. The red line shows that the attribute will be added to the end of the list.

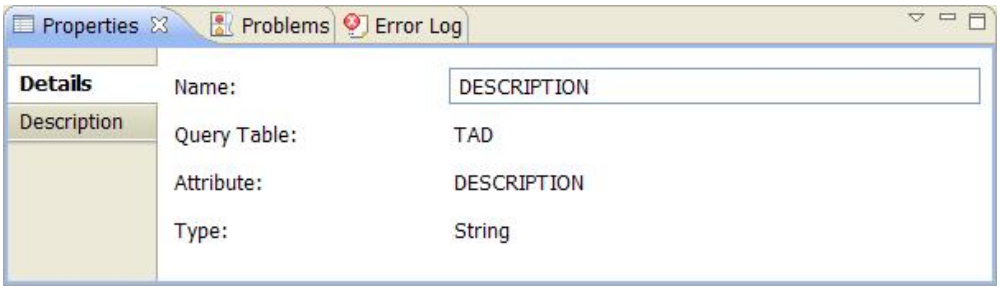
You can change the order of the attributes in the attribute table by selecting an attribute and dropping it on its new position in the table.

To delete an attribute from the list, right-click the attribute and select **Delete** from the context menu. Alternatively, select the attribute and press the delete key (**DEL**) on your keyboard.

3.2.11. Changing the properties of an attribute

To show and edit the properties of an attribute, select the attribute and go to the **Properties** view.

The Properties view of an attribute consists of the **Details** and the **Description** tabs. The **Detail** tab looks like this:

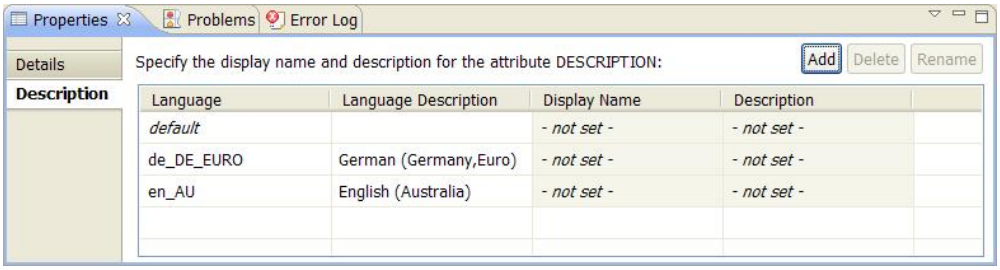


The following properties are available in the Properties view:

Property Name	Editable	Description
Name	Yes	The name of the attribute. This name must be in uppercase and unique within the query table.
Query Table	No	The short name of the query table from which the attribute was added.
Attribute	No	The original name of the added attribute.
Type	No	The type of the attribute.

Note: The Properties view for attributes of the primary or attached query tables contains only the **Name** and **Type** properties, both of which cannot be edited.

The **Description** tab looks like this:



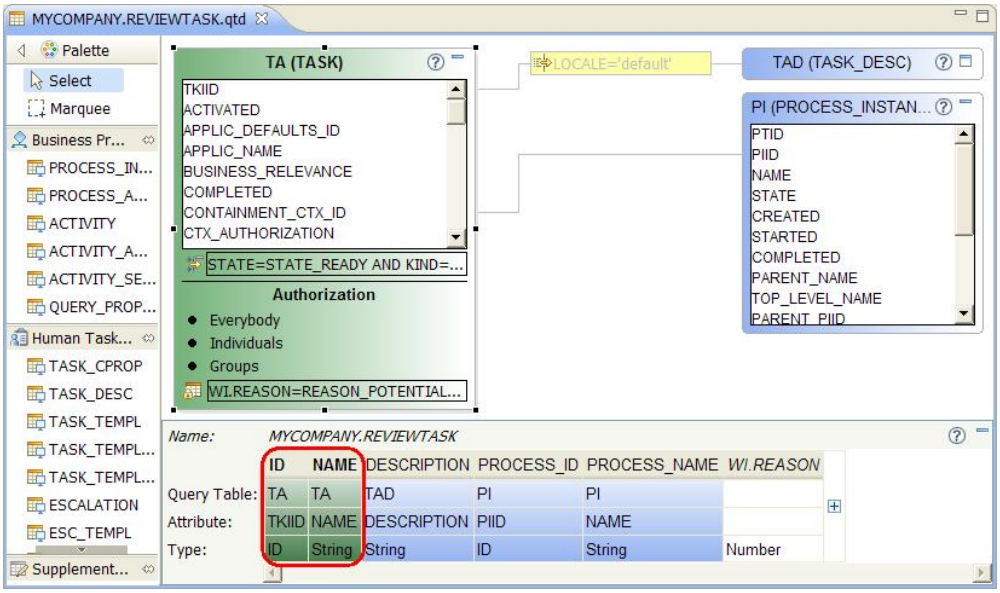
This view lists the languages (refer to *Localization* on page 32 for details) that have been defined for this query table, as well as for each language the display name and the description for the currently selected attribute. Initially, this view contains only the **default** language.

Refer to *Creating, modifying and deleting a language* on page 37 on how to work with languages.

3.2.12. Visualizing references to attributes

To facilitate the management of query tables, the Query Table Builder visualizes from which primary or attached query table the attributes in the attributes table originate.

When a primary or an attached query table is selected, all of its attributes in the attributes table are highlighted. The following figure shows an example:



In this example, the primary query table **TA** is selected, and the two attributes **ID** and **NAME** are highlighted in the attributes table.

Similarly, when an attribute is selected in the attributes table, the primary or attached query table from which it originates is highlighted, and its attribute list is positioned to show the original attribute.

3.2.13. Creating a composite query table for Business Space for Human Workflow

The Business Space for Human Workflow is able to use query tables deployed on Business Process Manager in order to show lists in the widgets **Tasks**, **Processes**, **Escalations**, **Task Definitions, Process Definitions** and **Work Baskets**. By creating query tables with different filters, it is therefore possible to provide customized lists for Business Space containing certain tasks, processes or definitions only. Depending on the attributes defined in these query tables, these lists can display different information, including business data.

When displaying lists defined in query tables, the Business Space uses language information that has been deployed along with the query table (refer to *Localization* on page 32 for details). Depending on the language settings of Business Space, it is thus possible to view localized display names and descriptions.

Query tables that will be used in Business Space have to fulfill certain criteria:

*Query tables for the widget **Tasks**:*

1. The primary query table has to be **TASK**.
2. The attributes **KIND** and **STATE** of the primary query table TASK have to be added to the query table. To enable drill-down scenarios that include the tasks associated with a process instance, add the attributes **CONTAINMENT_CTX_ID** and **TKTID**. Additionally, for displaying details in the Tasks List it is recommended to add **IS_WAIT_FOR_SUB_TK**.

*Query tables for the widget **Processes**:*

1. The primary query table has to be **PROCESS_INSTANCE**.
2. The attributes **PTID**, **STATE**, **PARENT_PIID** and **TOP_LEVEL_PIID** of the primary query table PROCESS_INSTANCE have to be added to the query table.

*Query tables for the widget **Escalations**:*

1. The primary query table has to be **ESCALATION**.
2. The attributes **STATE** and **KIND** of the primary query table ESCALATION have to be added to the query table.

*Query tables for the widget **Task Definitions**:*

1. The primary query table has to be **TASK_TEMPL**.
2. An attached query table must be **PROCESS_TEMPLATE**.
3. The attributes **VALID_FROM**, **KIND** and **STATE** of the primary query table TASK_TEMPL have to be added to the query table.
4. The attributes **PTID** and **STATE** of the attached query table PROCESS_TEMPLATE have to be added to the query table.

*Query tables for the widget **Process Definitions**:*

1. The primary query table has to be **PROCESS_TEMPLATE**.
2. The attributes **VALID_FROM**, **EXECUTION_MODE** and **STATE** of the primary query table PROCESS_TEMPLATE have to be added to the query table.

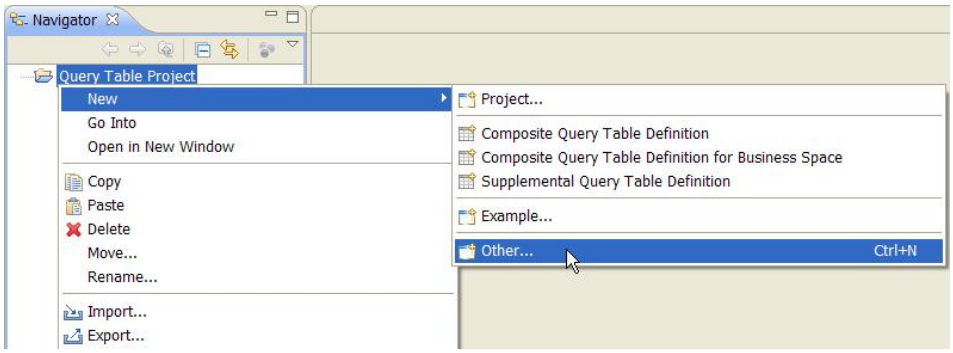
*Query tables for the widget **Work Baskets**:*

1. The primary query table has to be **WORK_BASKET**.
2. The attributes **NAME** and **WBID** of the primary query table WORK_BASKET have to be added to the query table.

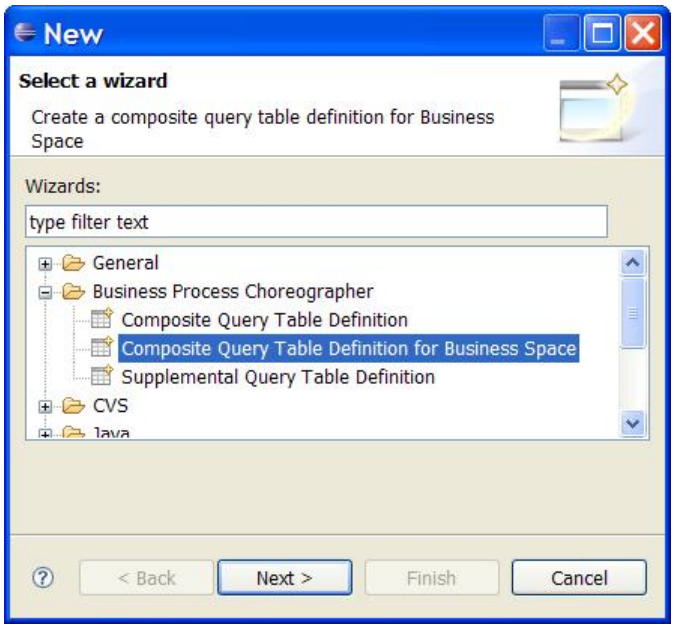
The Query Table Builder provides the option to create a new query table explicitly for use in Business Space. When using this feature, the created query table automatically contains the correct primary query table, the correct attached query table as well as the required attributes so that it can be used by Business Space.

To create a composite query table for Business Space, complete the following steps.

1. Switch to the **Navigator**, right-click the project in which the query table definition file is to be created, and select **New > Other**.

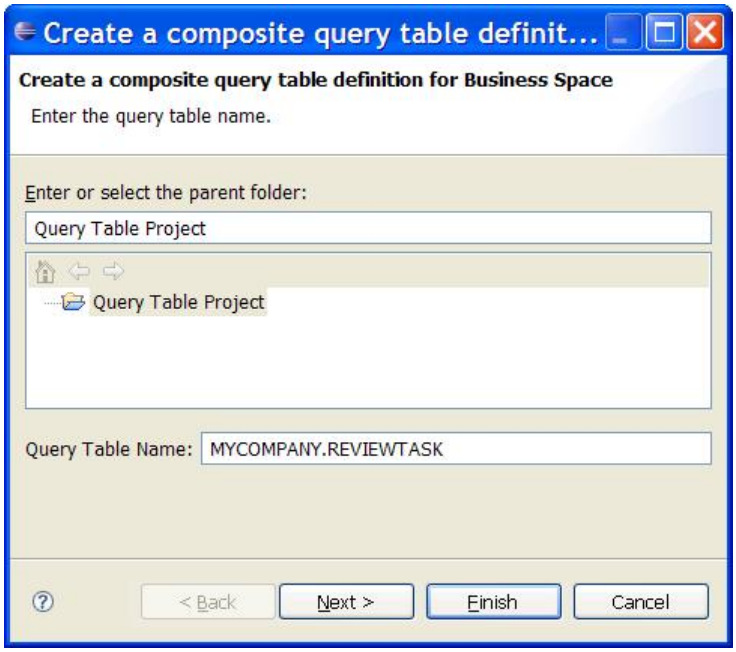


2. In the window that opens, specify the type of query table that is to be created:



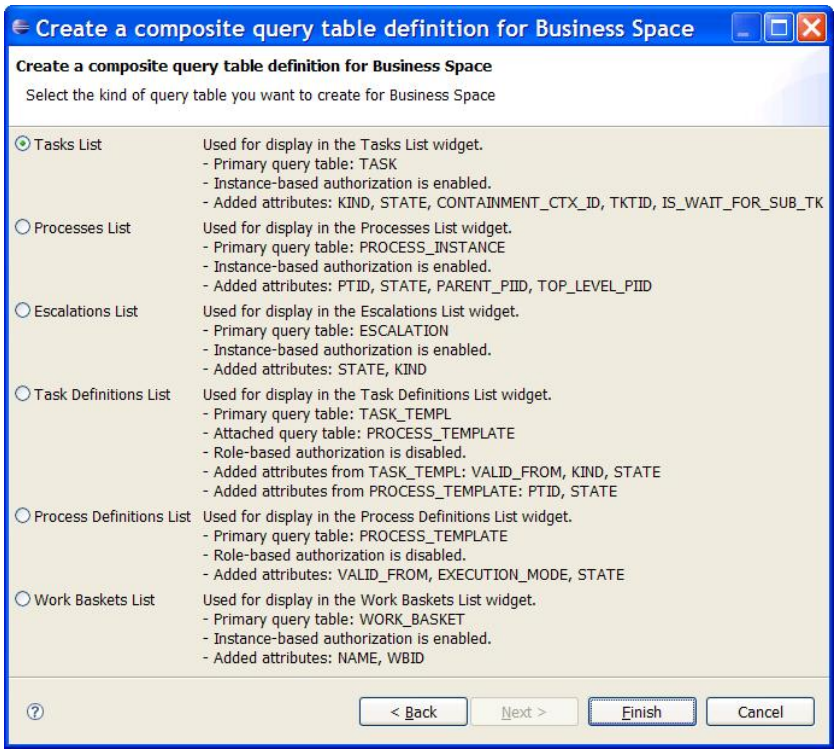
In the **Business Process Choreographer** section, select **Composite Query Table Definition for Business Space,** and click **Next**.

3. Select the parent folder for the query table definition and the name of the query table:



In the **Query Table Name** field, enter the name of the new query table. The name must be in uppercase and have the format *PREFIX.NAME*. It can have a maximum of 28 characters. The prefix and the name can contain numbers, but they cannot start with one. You cannot change the name of the query table after the query table definition file is created.

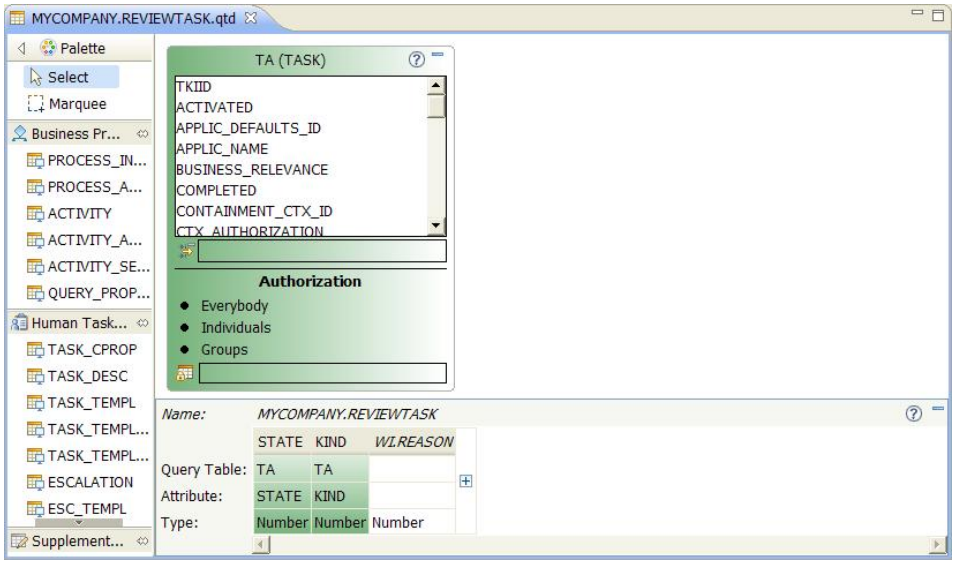
4. Click **Next**. The following wizard page appears:



Select the appropriate kind of query table you want to create and click **Finish**.

The composite query table is created, and the editor opens the new query table definition file. In addition to the query table definition .qtd file, a file with the extension .qtdex is also created. This file stores the layout information for the query table.

The initial view of the new query table looks similar to the following figure:



As shown, the created query table has the appropriate primary query table – in this case **TASK** for the widget Tasks List – and contains the required attributes **STATE** and **KIND**. The composite query table can now be edited as described in the previous sections.

3.3. Working with supplemental query tables

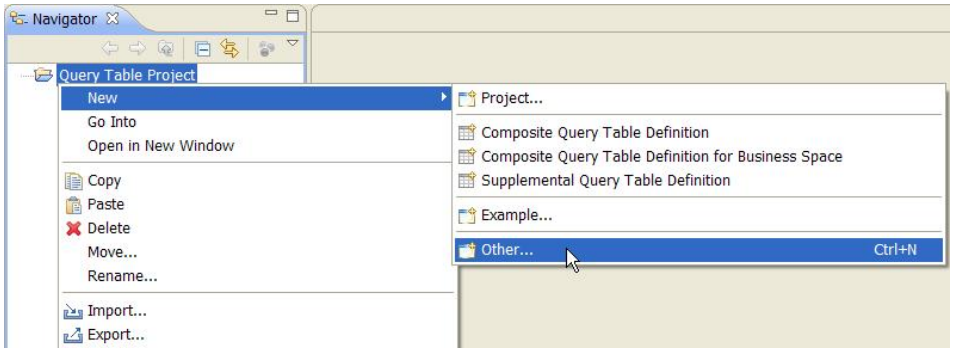
This section describes how to create and edit supplemental query tables in the Query Table Builder. Supplemental query tables can be defined to use custom database tables colocated in the Business Process Choreographer database.

3.3.1. Creating a supplemental query table

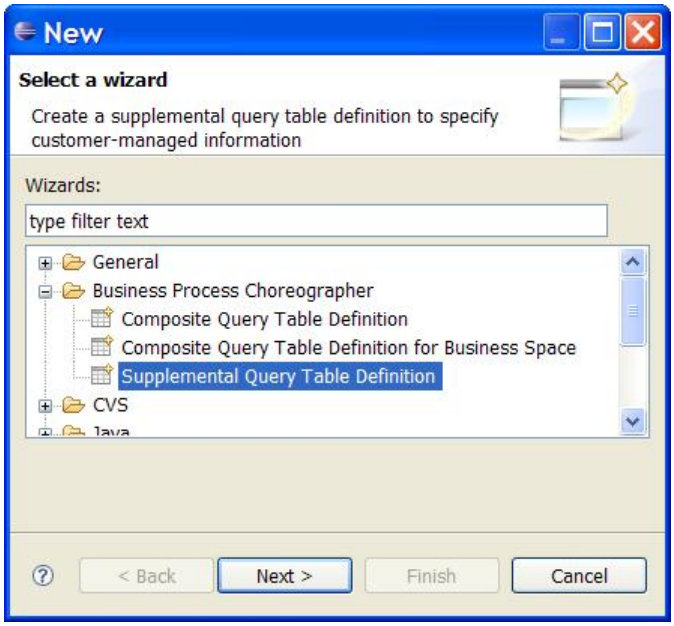
You can create a query table definition (.qtd) file in any type of project, for example, a simple project, or a Java project. However, you can put a query table definition only in the root directory of a project, not in any subdirectories or packages.

To create a supplemental query table, complete the following steps.

1. Switch to the **Navigator**, right-click the project in which the query table definition file is to be created, and select **New > Other**.

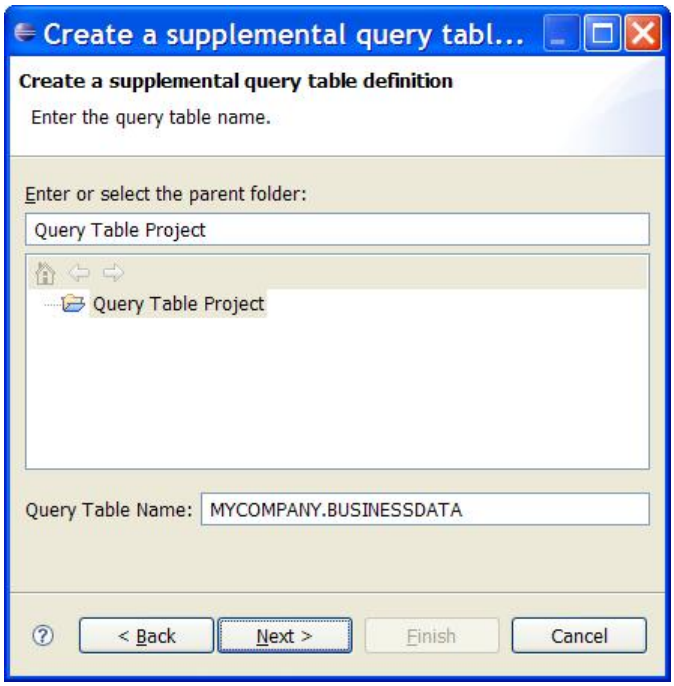


2. In the window that opens, specify the type of query table that is to be created:



In the **Business Process Choreographer** section, select **Supplemental Query Table Definition,** and click **Next**.

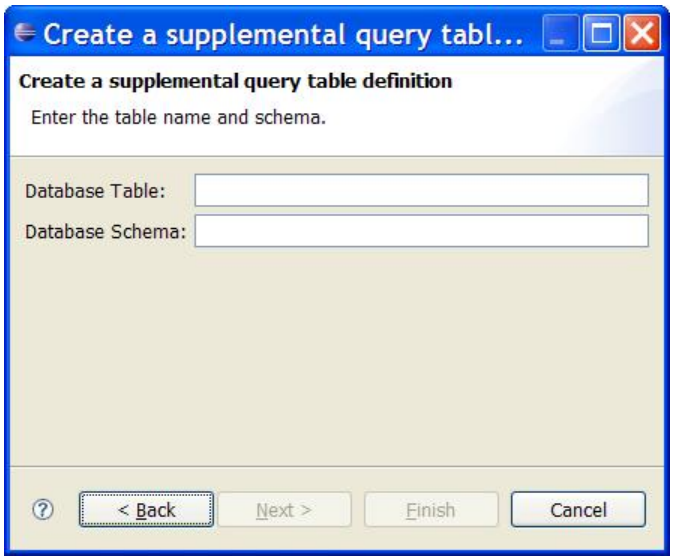
3. Select a parent folder for the query table definition and the name of the query table:



In the **Query Table Name** field, enter the name of the new query table. The name must be in uppercase and have the format *PREFIX.NAME*. It can have a maximum of 28 characters. The prefix and the name can contain numbers, but they cannot start with one. You cannot change the name of the query table after the query table definition file is created.

4. Click **Next**.

5. Specify the **Database Table** and the **Database Schema**:



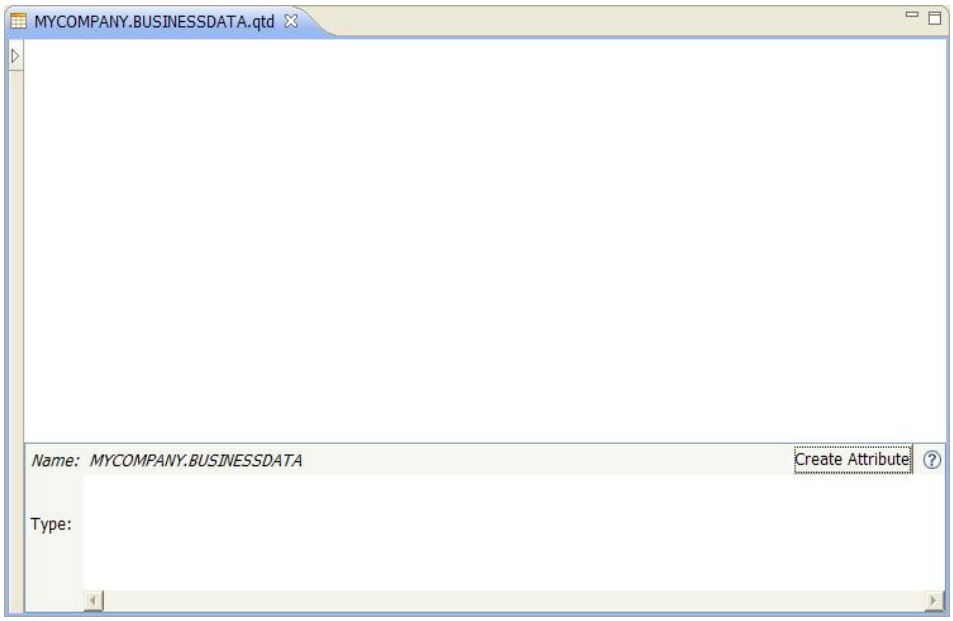
The values specified here can be changed after the supplemental query table is created.

6. Click **Finish.**

   The supplemental query table is created, and the editor opens the new query table definition file. In addition to the query table definition .qtd file, a file with the extension .qtdex is also

created. This file stores the layout information for the query table.

The initial view of the new query table looks similar to the following figure:



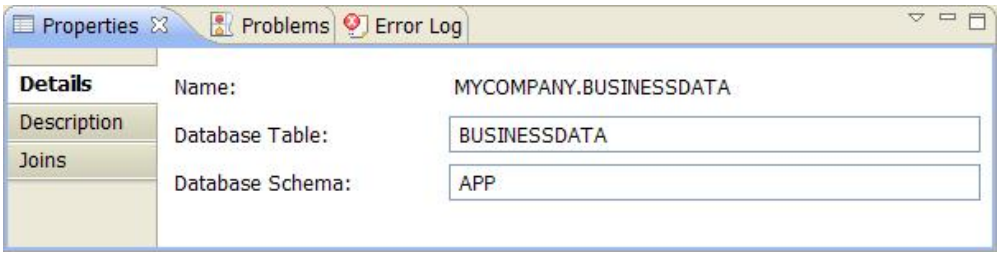
While editing a supplemental query table, the palette and the canvas are not used. Only the attributes table and the Properties view are required.

If you are editing a composite query table, the new supplemental query table is also added to the palette.

3.3.2. Changing the properties of the query table

To show and edit the properties of the supplemental query table, select either the canvas or the attributes table and go to the **Properties** view.

The Properties view of the query table consists of the **Details**, the **Description**, and the **Joins** tabs. The **Details** Properties view looks like this:

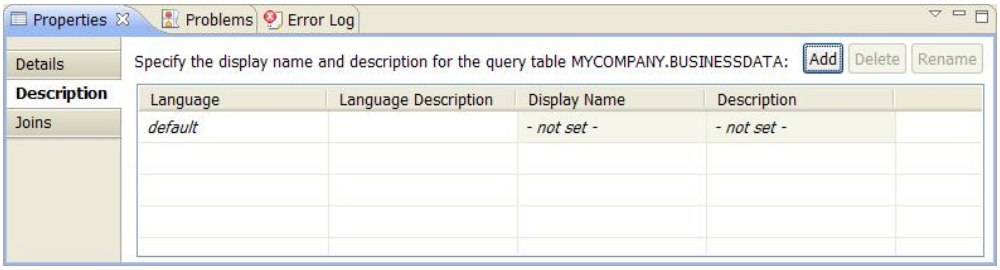


The following properties are available in the **Details** Properties view:

Property Name	Editable	Description
Name	No	The name of the supplemental query table. This is the name that was specified when the query table was created. You cannot change the name after the query table is created.
Database Table	Yes	The name of the database table for which this supplemental query table is created.
Database Schema	Yes	The name of the database schema for which this supplemental query table is created.

The name of the query table is also displayed at the top of the attributes table.

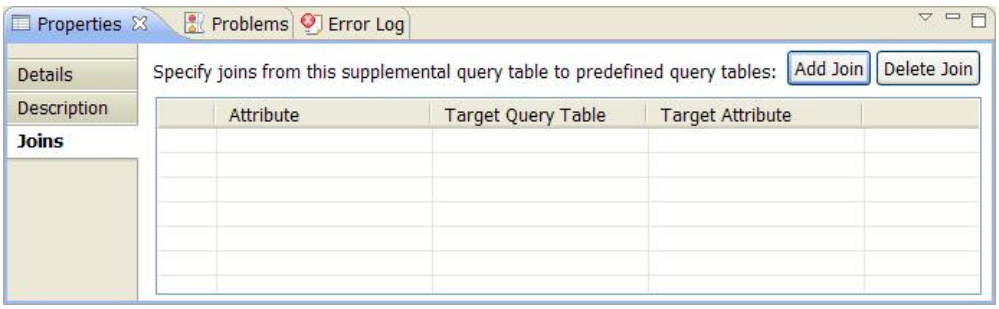
The **Description** Properties view looks like this:



This view lists the languages (refer to *Localization* on page 32 for details) that have been defined for this query table, as well as for each language the display name and the description for the query table. Initially, this view contains only the **default** language.

Refer to *Creating, modifying and deleting a language* on page 37 on how to work with languages.

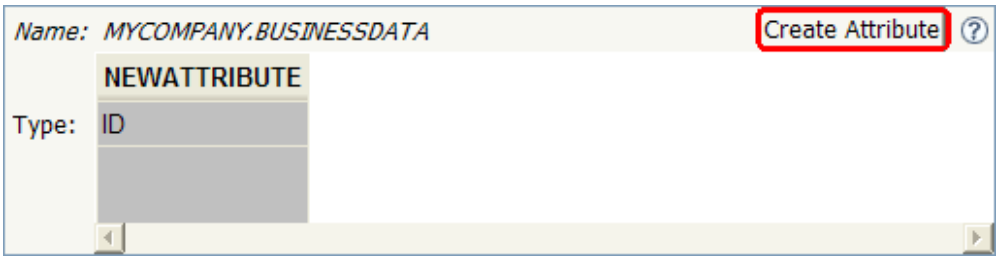
The **Joins** Properties view looks like this:



This view lists all of the joins that are defined for this supplemental query table. How to create, edit, and delete joins is described in *Creating and deleting a join* on page 59.

3.3.3. Adding attributes to and removing attributes from the query table

To create a new attribute for the supplemental query table, click **Create Attribute**. This creates a new attribute and adds it to the end of the attributes list:

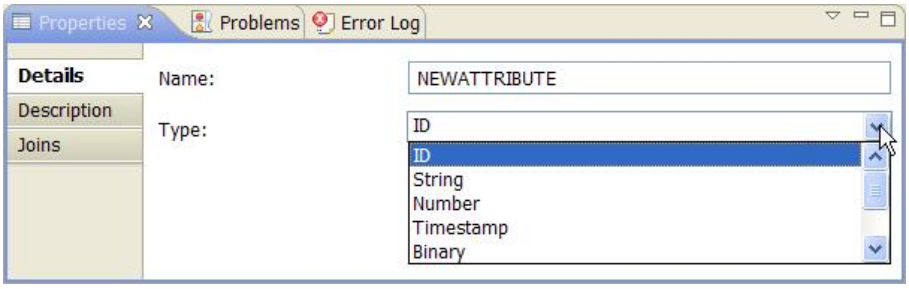


The attributes in the attributes table can also be reordered and deleted as described in *Working with the attributes table* on page 46 for the composite query table.

3.3.4. Changing the properties of an attribute

To show and edit the properties of an attribute, select it and go to the **Properties** view.

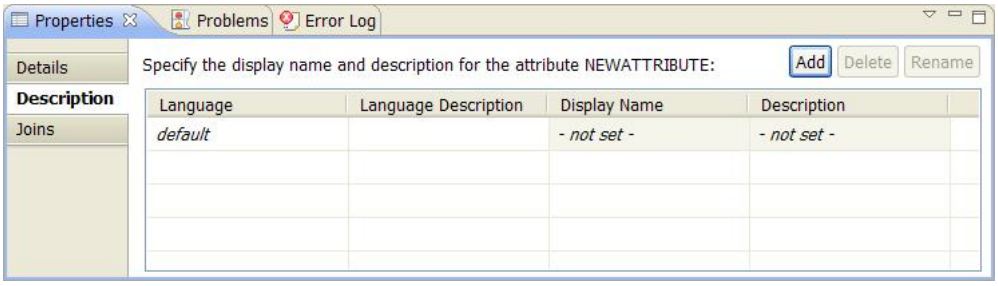
The Properties view of an attribute consists of the **Details**, the **Description**, and the **Joins** tabs. The **Details** tab looks like this:



The following properties are available in the **Details** Properties view:

Property Name	Editable	Description
Name	Yes	The name of the attribute. This name must be in uppercase and unique within the query table. The name must be the name of an existing column in the referenced database table.
Type	Yes	The type of the attribute, which can be selected from the drop-down box. The type must be compatible with the database type of the referenced column in the database table.

The **Description** Properties view looks like this:



This view lists the languages (refer to *Localization* on page 32 for details) that have been defined for this query table, as well as for each language the display name and the description for the currently selected attribute. Initially, this view contains only the **default** language.

Refer to *Creating, modifying and deleting a language* on page 37 on how to work with languages.

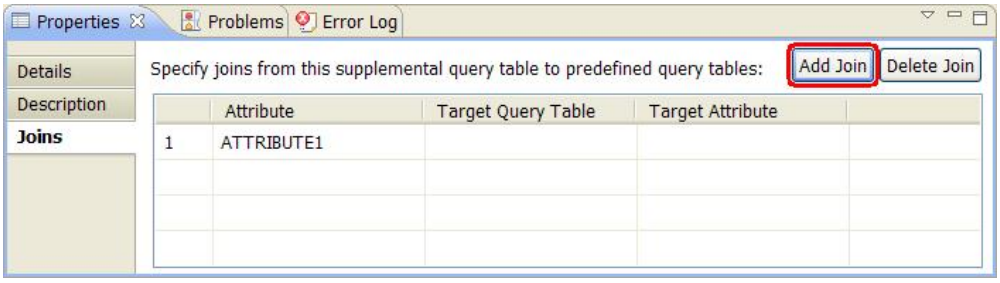
The **Joins** Properties view is identical to the one shown in *Changing the properties of the query table* on page 56 for the entire supplemental query table. However, this view lists only those joins that are defined for this specific attribute.

3.3.5. Creating and deleting a join

A join has the following properties:

Property Name	Description
Attribute	The attribute of the supplemental query table that is the source of the join. Any attribute that is defined in the attributes table can be specified in this property.
Target Query Table	The predefined query table to which the join refers.
Target Attribute	The attribute of the predefined query table to which the join refers to.

To create a join, click **Add Join** in the **Joins** tab of the Properties view. A join is created and added to the table.



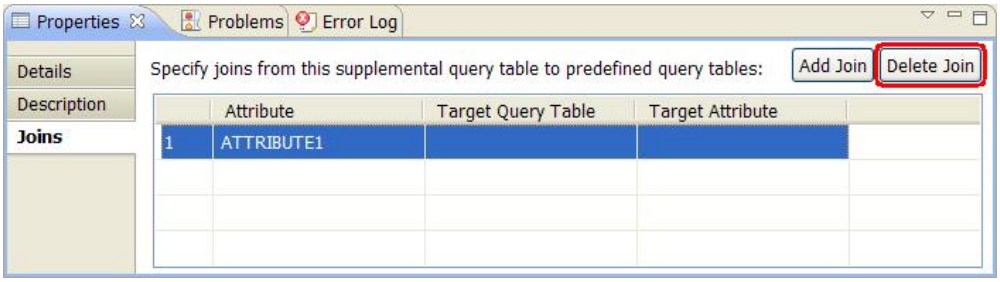
If the join is created in the Properties view of a selected attribute, as in this example, the join is created with its **Attribute** property

already set. In this case, you cannot change the Attribute property.

If the join is created in the Properties view of the supplemental query table, the **Attribute** property is not set. In this case, you can set the attribute by selecting an attribute from the list in the cell of the join property.

Editing of the **Target Query Table** and **Target Attribute** is the same as for the Properties view of a selected attribute of the supplemental query table. In both cases, clicking in the cell of the join property opens a drop-down box with the available values.

To delete a join, select the join row. To do so, click either in the first column that contains the number of the join, or in the empty column next to **Target Attribute**. The row will be highlighted:



To delete the highlighted join, click **Delete Join**.

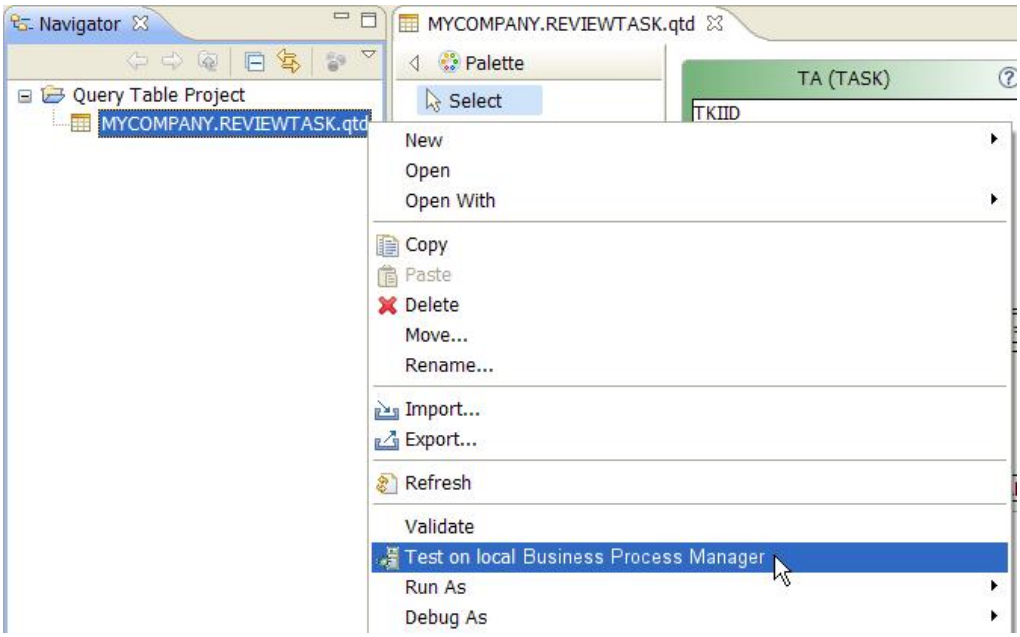
3.4. Administering query tables

The Query Table Builder contains a test client for deploying, undeploying, updating, listing and executing query tables on a locally installed Business Process Manager 7.5 profile.

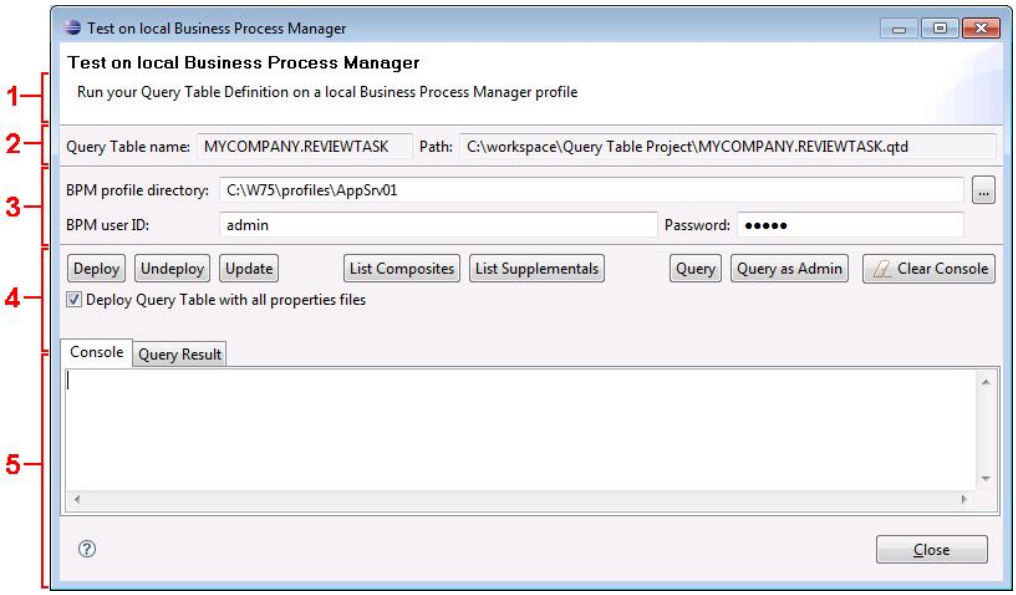
This test client uses the jython script for the corresponding commands:

```
<BPM install directory>/ProcessChoreographer/admin/Ã
                                                manageQueryTables.py
```

To start the test client, right-click on an existing query table definition file in the **Navigator** and select **Test on local Business Process Manager** from the context menu.



The test client opens.



1. Message area: If any of the entered information is invalid, an error message is displayed here.

2. Query table definition information area

   a. Query Table Name: The name of the query table is used when executing one of the administrative commands.

   b. Path: Absolute path on your local drive to the .qtd file. This is the file that is passed to Business Process Manager for the commands deploy, undeploy, and update.

3. Configuration area: All values, except for the password, are stored and prefilled when the test client is started again.

   a. BPM profile directory: Enter the absolute path to the Business Process Manager 7.5 profile directory on your local computer.
   Click **Browse** (...) to browse through the local directories interactively.

   b. BPM user ID: Enter a user ID that has the authority to execute administrative commands. You can leave this field blank if you want to interactively enter the user ID and password when executing the commands.

   c. Password: Enter the password for the BPM user ID. This password is stored in memory during the execution of the Integrated Development Environment, but discarded when exiting.

4. Command area

   a. Command buttons: The commands are explained in detail below.

b. Deploy option: If checked, the current query table definition is deployed or updated together with all corresponding property files.
If unchecked, only the query table definition is deployed or updated.

5. Output area

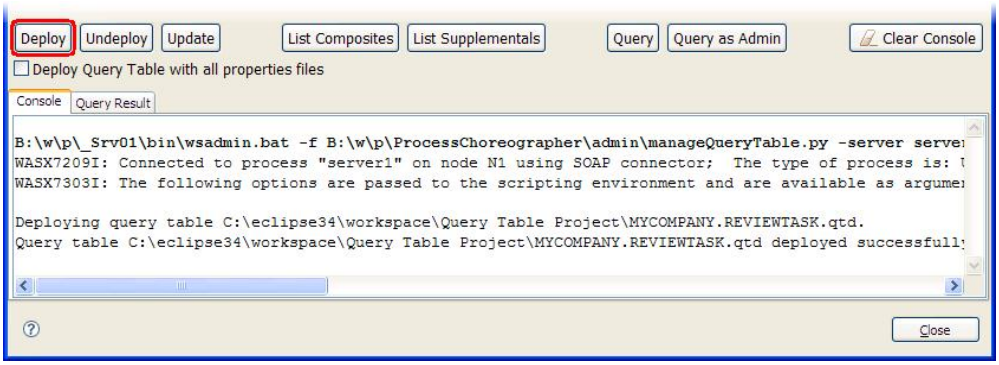
a. Console: The console shows the commands that were executed and their output.

b. Query Result: If you run the query command, the output of the query is shown in a table.

3.4.1. Deploying query table definitions

You must deploy a query table definition (qtd) file, to the Business Process Manager profile before you can run the query. To deploy the query table definition for the first time, click **Deploy** in the test client. If you want to replace a deployed .qtd file, use the update command.

The following figure shows sample output produced by the deploy command. After each run, review the output for error messages.



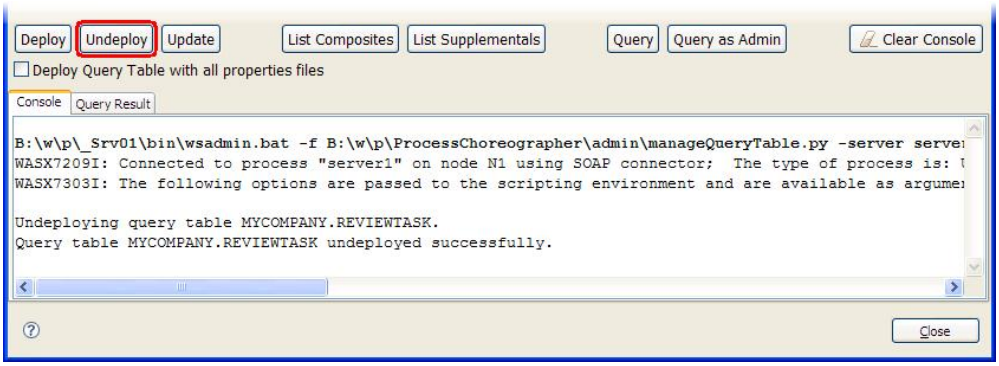
The deploy command has the following syntax:

```
<BPM profile directory>/bin/wsadmin –f
<BPM install directory>/ProcessChoreographer/admin/Ã
                                        manageQueryTable.py
     -server <servername>
     -user <BPM user id>
     -password <password>
     -deploy <path to qtd or jar file>
```

3.4.2. Undeploying query table definitions

To remove unused query table definitions from a Business Process Manager profile, click **Undeploy**. You can undeploy the current query table definition only.

The following figure shows sample output produced by the undeploy command. After each run, review the output for error messages.



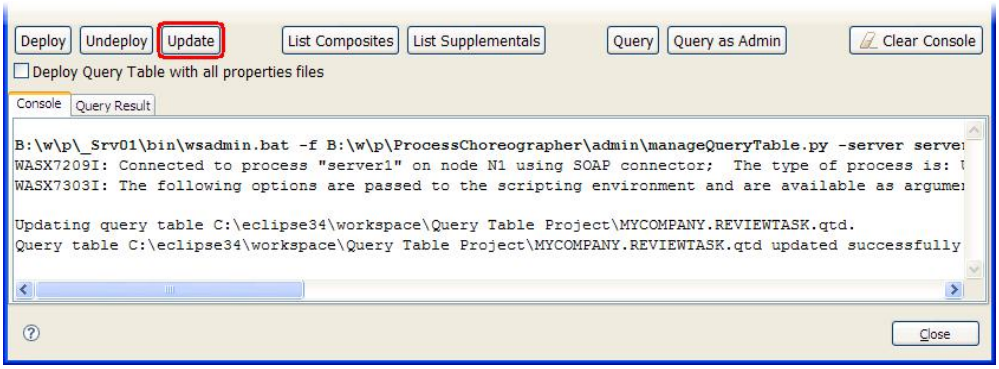
The undeploy command has the following syntax:

```
<BPM profile directory>/bin/wsadmin –f
<BPM install directory>/ProcessChoreographer/admin/Ã
                                        manageQueryTable.py
     -server <servername>
     -user <BPM user id>
     -password <password>
     -undeploy <query table name>
```

3.4.3. Updating a deployed query table definition

If you modify a .qtd file that is already deployed, you need to update the query table definition before you can run the query. Click **Update** to replace the query table definition file.

The following figure shows sample output produced by the update command. After each run, review the output for error messages.

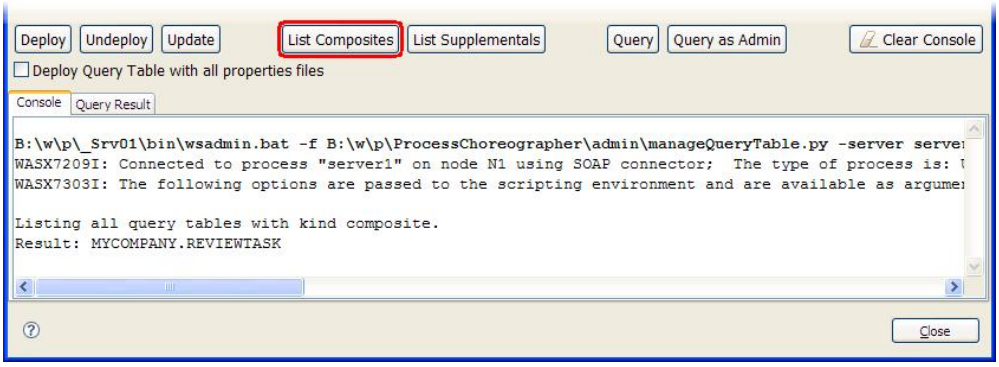


The update command has the following syntax:

```
<BPM profile directory>/bin/wsadmin –f
<BPM install directory>/ProcessChoreographer/admin/Ã
                                        manageQueryTable.py
     -server <servername>
     -user <BPM user id>
     -password <password>
     -update definition <path to qtd or jar file>
```

3.4.4. Listing composite and supplemental query tables

To get an overview of the deployed composite and supplemental query tables, click either **List Composites** or **List Supplementals**. The following figure shows sample output produced by the List Composites command. After each run, review the output for error messages.



The following command is executed for listing composite query tables:

```
<BPM profile directory>/bin/wsadmin –f
<BPM install directory>/ProcessChoreographer/admin/Ã
                                        manageQueryTable.py
     -server <servername>
     -user <BPM user id>
     -password <password>
     -query names –kind composite
```

For supplemental query tables, the last parameter above is `supplemental` instead.

3.4.5. Running the query

The Query Table Builder contains a Java 2 Enterprise Edition Application Client (query client) that uses the BusinessFlow-Manager EJB. For further information on this EJB, refer to http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.wbpmcore.javadoc.doc/web/apidocs/com/ibm/bpe/api/BusinessFlowManager.html

The query client uses the method `queryRows(java.lang.String queryTableName, FilterOptions filterOptions, AuthorizationOptions authorizationOptions, java.util.List parameters)` with a threshold of 50 rows. While null is passed as the value of `parameters`, the value of `authorizationOptions` depends on the query functionality that is being used.

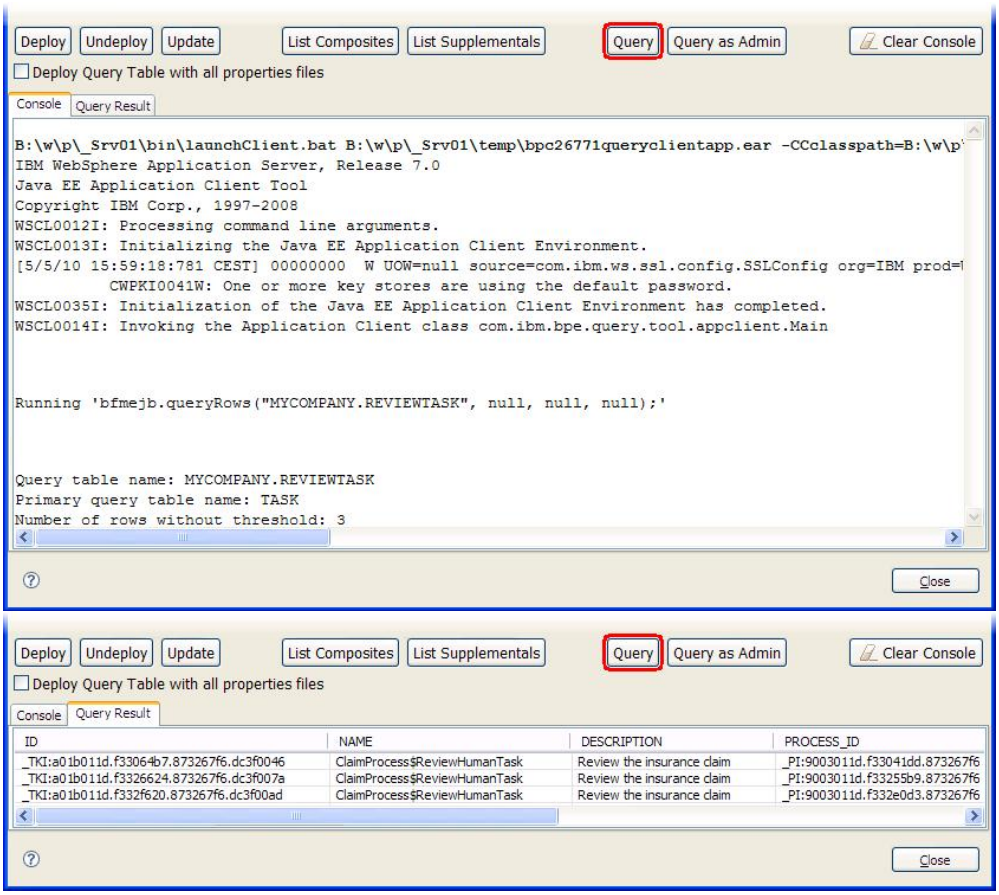
The test client provides two different query modes:

- Query: The query is executed using null as the value of `authorizationOptions`. In this case, the values defined in the query table definition are used.

- Query as Admin: The query is executed using `AdminAuthorizationOptions` as the value of `authorizationOptions`. As a consequence, the query is performed for a system administrator.

Note: Make sure that Business Process Manager contains data for the filters and conditions of the query table definition, otherwise the query client cannot display any sample output.

The following figure shows the command output and the query result. After each run, and especially when no data is returned, review the output for error messages.


Deploy
Undeploy
Update
List Composites
List Supplementals
Query
Query as Admin
Clear Console
Deploy Query Table with all properties files
Console
Query Result
B:\w\p_Srv01\bin\launchClient.bat B:\w\p_Srv01\temp\bpc26771queryclientapp.ear -CCclasspath=B:\w\p
IBM WebSphere Application Server, Release 7.0
Java EE Application Client Tool
Copyright IBM Corp., 1997-2008
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the Java EE Application Client Environment.
[5/5/10 15:59:18:781 CEST] 00000000 W UOW=null source=com.ibm.ws.ssl.config.SSLConfig org=IBM prod=
CWPKI0041W: One or more key stores are using the default password.
WSCL0035I: Initialization of the Java EE Application Client Environment has completed.
WSCL0014I: Invoking the Application Client class com.ibm.bpe.query.tool.appclient.Main
Running 'bfmejb.queryRows("MYCOMPANY.REVIEWTASK", null, null, null);'
Query table name: MYCOMPANY.REVIEWTASK
Primary query table name: TASK
Number of rows without threshold: 3
Close

Deploy
Undeploy
Update
List Composites
List Supplementals
Query
Query as Admin
Clear Console
Deploy Query Table with all properties files
Console
Query Result
ID
NAME
DESCRIPTION
PROCESS_ID
_TKI:a01b011d.f33064b7.873267f6.dc3f0046
ClaimProcess$ReviewHumanTask
Review the insurance claim
_PI:9003011d.f33041dd.873267f6
_TKI:a01b011d.f3326624.873267f6.dc3f007a
ClaimProcess$ReviewHumanTask
Review the insurance claim
_PI:9003011d.f33255b9.873267f6
_TKI:a01b011d.f332f620.873267f6.dc3f00ad
ClaimProcess$ReviewHumanTask
Review the insurance claim
_PI:9003011d.f332e0d3.873267f6
Close

4. Exporting and importing query table definitions

A query table definition consists of the following files:

- .qtd file: The query table definition itself without language dependent content.
- .qtdex file: A file used by the Query Table Builder to store layout information in Eclipse. This file is not used by Business Process Manager.
- .properties files: One or more files containing language dependent content.

Depending on the intention when exporting query table definitions, a different set of file types is needed. Here are some common examples:

- Deployment on Business Process Manager: Business Process Manager supports both modes – deploying one .qtd file only, or deploying one or multiple .qtd files, optionally with language dependent display names and descriptions. For the first mode, only export the .qtd files you want to deploy into a directory. For the second mode, export all .qtd files and .properties files of all query table definitions which you want to deploy within one step into a jar file.
- Exchange query table definitions with other developers: In this case, you should export all .qtd, .qtdex and .properties files you want to share into a jar file or a directory.
- Sending files to translation bureaus:
  Translation bureaus work with .properties files only. Therefore, you should export .properties files into a directory for this intend.

Note: If you modify property files outside Eclipse, e.g. when working with translation bureaus, comply to the file encoding rules for Java property resource bundles.

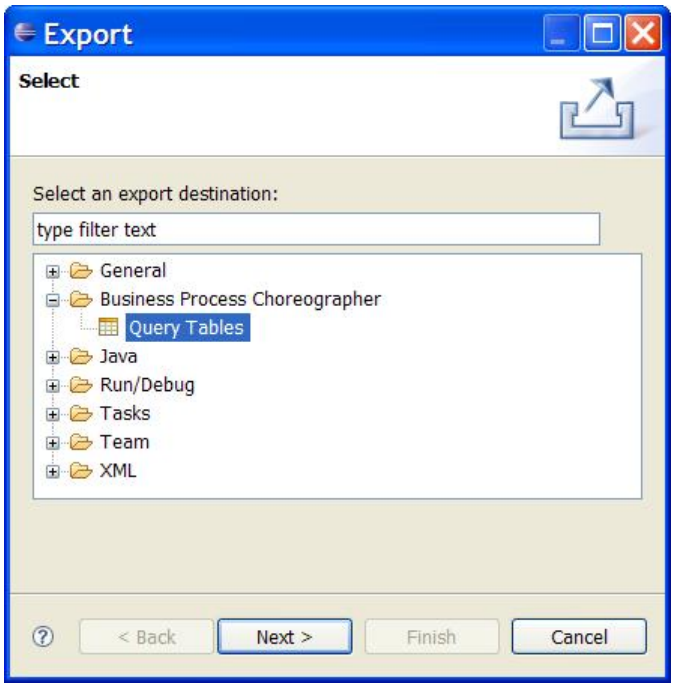
You can find them in the description for the Java class `java.util.PropertyResourceBundle`, which you can find either in your Java SDK documentation or online at http://java.sun.com/javase/6/docs/api/java/util/PropertyResourceBundle.html.

4.1. Exporting

To export the files, use the export wizard **Query Tables**:

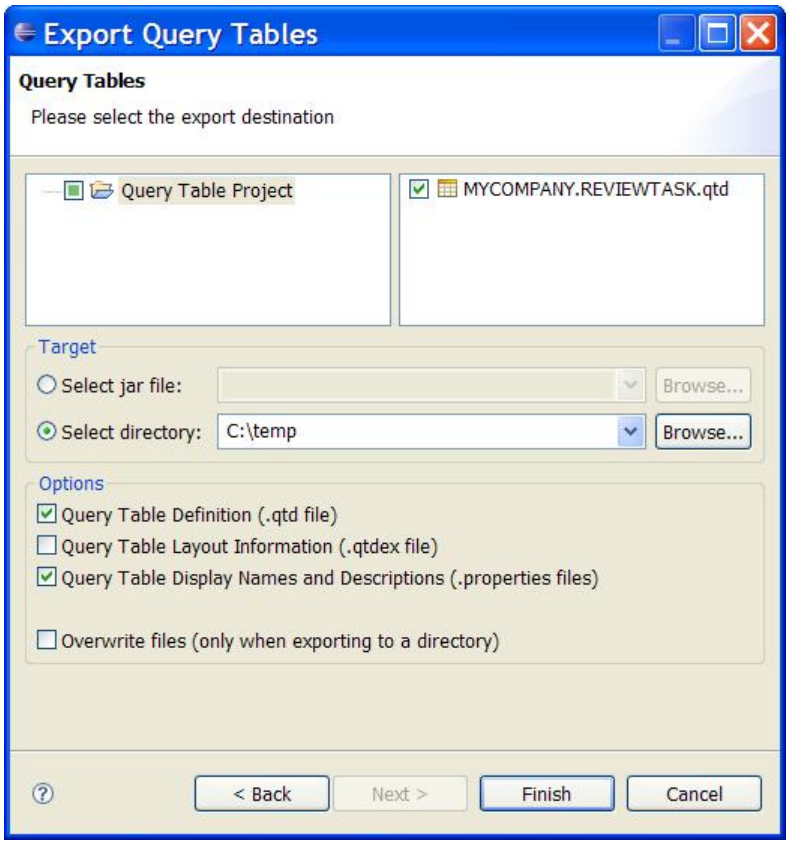
1. In the Eclipse menu, select **File > Export...**

2. In the export dialog, expand category **Business Process Choreographer**, select the export wizard **Query Tables** and click on **Next**.



3. Select all query table definition files you need to export in one step and specify either a JAR file or a directory where you want to export the files to.

4. Click on **Finish** to export the qtd files.



Now you can transfer the exported files to the target system.

4.2. Importing

You may want to import query table definitions into Eclipse for different reasons. Two of the most common scenarios are:

- Importing query table definitions which were previously exported by another developer.
- Importing .properties files for an existing query table definition which were provided by translation bureaus.

For importing query table definitions or related files into an existing project in Eclipse, you have different options:

Drag-and-drop

From a file browser which supports drag-and-drop on your operating system, select the files which belong to the query table definitions you want to import, i.e. .qtd, .qtdex and .properties files.

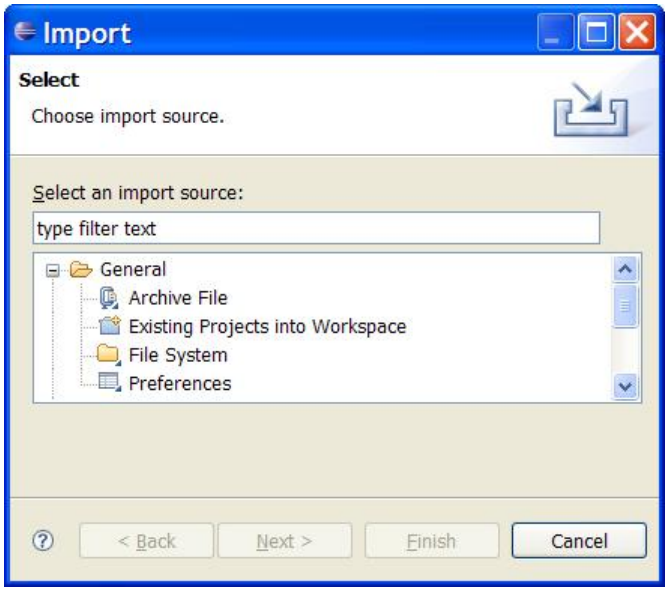
Eclipse import wizards

Use Eclipse provided import wizards to import query table definitions or translated .properties files into your workspace:

1. In the Eclipse menu, select **File > Import...**

2. In the import dialog, expand category **General**, select the appropriate import wizard and click on **Next**. If your import source is a directory, you should select the wizard **File**

System. For importing the content of a .jar file, select the wizard **Archive File**.



3. Select the source archive or directory where you want to import files from.
4. Select all files you want to import from the source.
5. Select the project into which you want to import the files.
6. Click **Finish** to start the import.

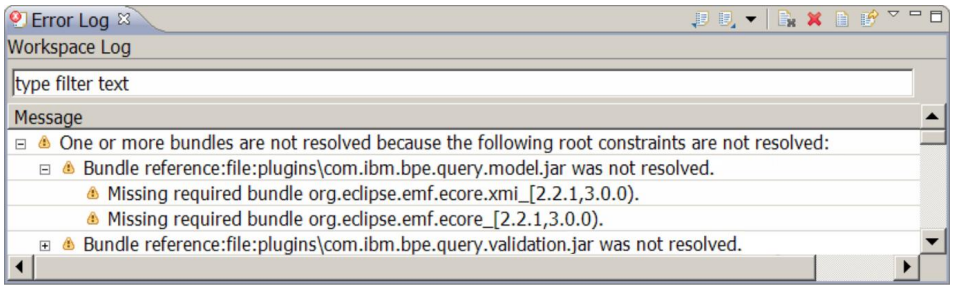
5. Troubleshooting

This section assists you in finding solutions to the most common issues that arise when using the Query Table Builder.

5.1. Installing the Query Table Builder

If you installed the Query Table Builder, but you cannot use any of the functionality offered, follow the steps listed in *Verifying the installation* on page 5. If you verified that the Query Table Builder is installed correctly, do the following:

1. Close Eclipse if it is running.
2. Start Eclipse with the following command line parameters:
   `eclipse -clean -debug`
3. In the Eclipse menu, select **Window > Show View > Other...**
4. In the dialog, expand section **General**, select **Error Log** and click on **OK**.
5. In the **Error Log** view, you get a list of unresolved bundles. Expand the items to see the cause why it could not be resolved:



In this example, you can see that EMF bundles are missing. In this case, you should either install the missing prerequisites or get one of the Eclipse distributions listed in *Integrated Development Environment* on page 1.

5.2. Working with files

The file name of a query table definition must not be changed. If you do this, you will get a number of unwanted behaviors:

- Query table definitions with relation among each others cannot be resolved, i.e. if one of your composite query table definitions references a supplemental query table definition and you rename the file name of the supplemental query table definition, the Query Table Builder will fail working on the composite query table definition.
- You only rename the .qtd file of the query table definition, but not the related .qtdex and .properties files, and

therefore you may loose information. Furthermore, the Query Table Builder might ask you if you want to delete these .qtdex and .properties files.

- When testing the query table definition in the built-in test client, the **Query** function will not work.

5.3. Using undo/redo

The Eclipse menus **Edit > Undo** and **Edit > Redo** are context sensitive, i.e. the operations offered by these menu items are dependent on which view is currently selected.

If you for example delete a query table definition and then work on another query table definition in the editor, the menu **Edit > Undo** can only undo operations in the editor. To undo the previous deletion of the query table definition, you first have to select the **Navigator** view and then select the menu **Edit > Undo**.