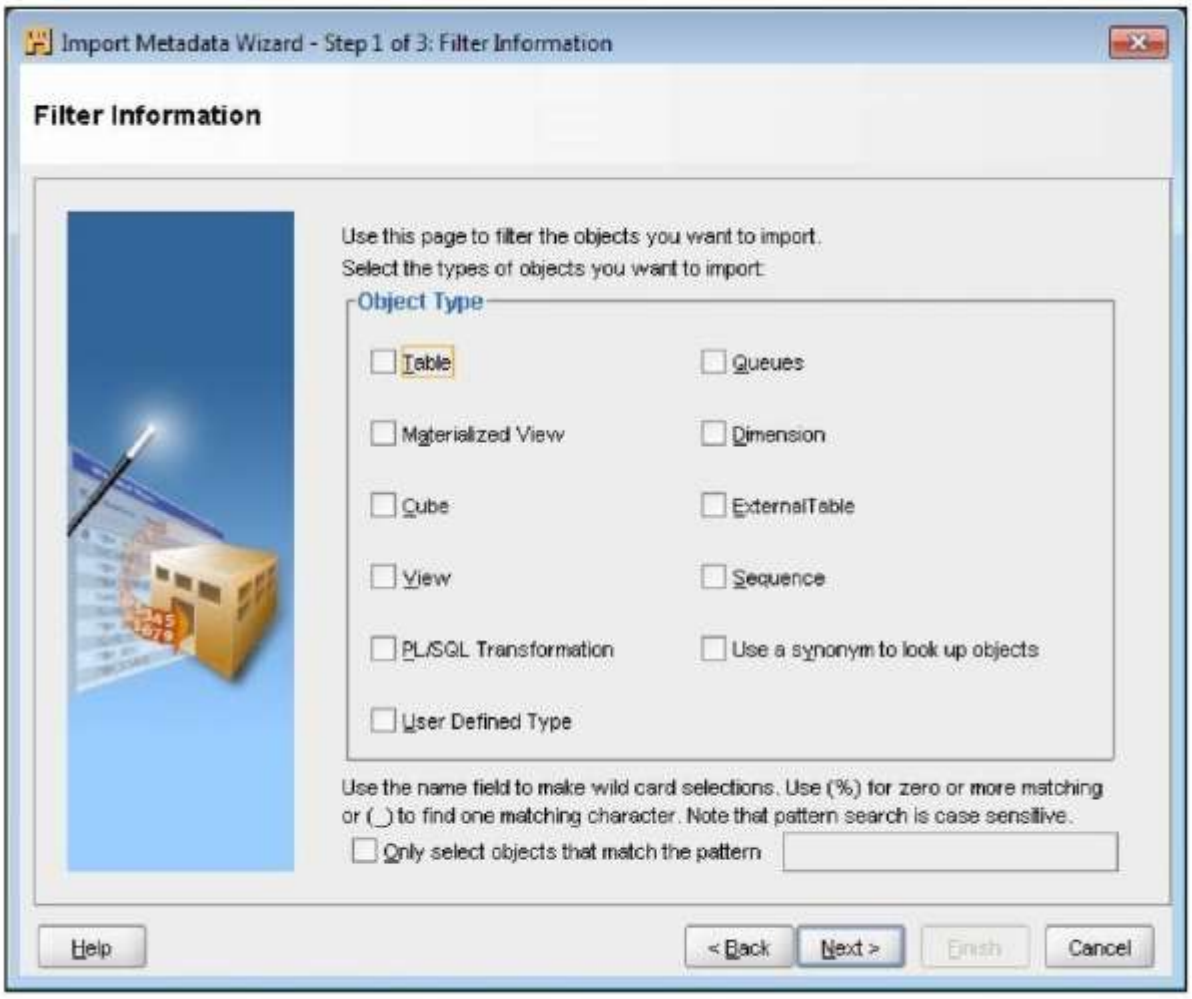
Importing source database objects from a database

We are now at the point where we can finally import our source database objects, source database objects. We'll walk through the process of importing from an Oracle database, which is very similar to the process of importing from a non-Oracle database; so it will be a good exercise to walk through pointing out specific differences as we go. After that, we will walk through the process of defining the metadata for our SQL Server database tables using a data object editor for tables, which is integrated into the Warehouse Builder for working with tables. We could just as easily import the database objects from SQL Server but we'll walk through the process for one source table manually just so we can learn about the process of editing data objects. We will then leave the rest as an exercise for the reader to be done in a similar manner or to be imported automatically. Let's start by importing from the Oracle database using the following steps:

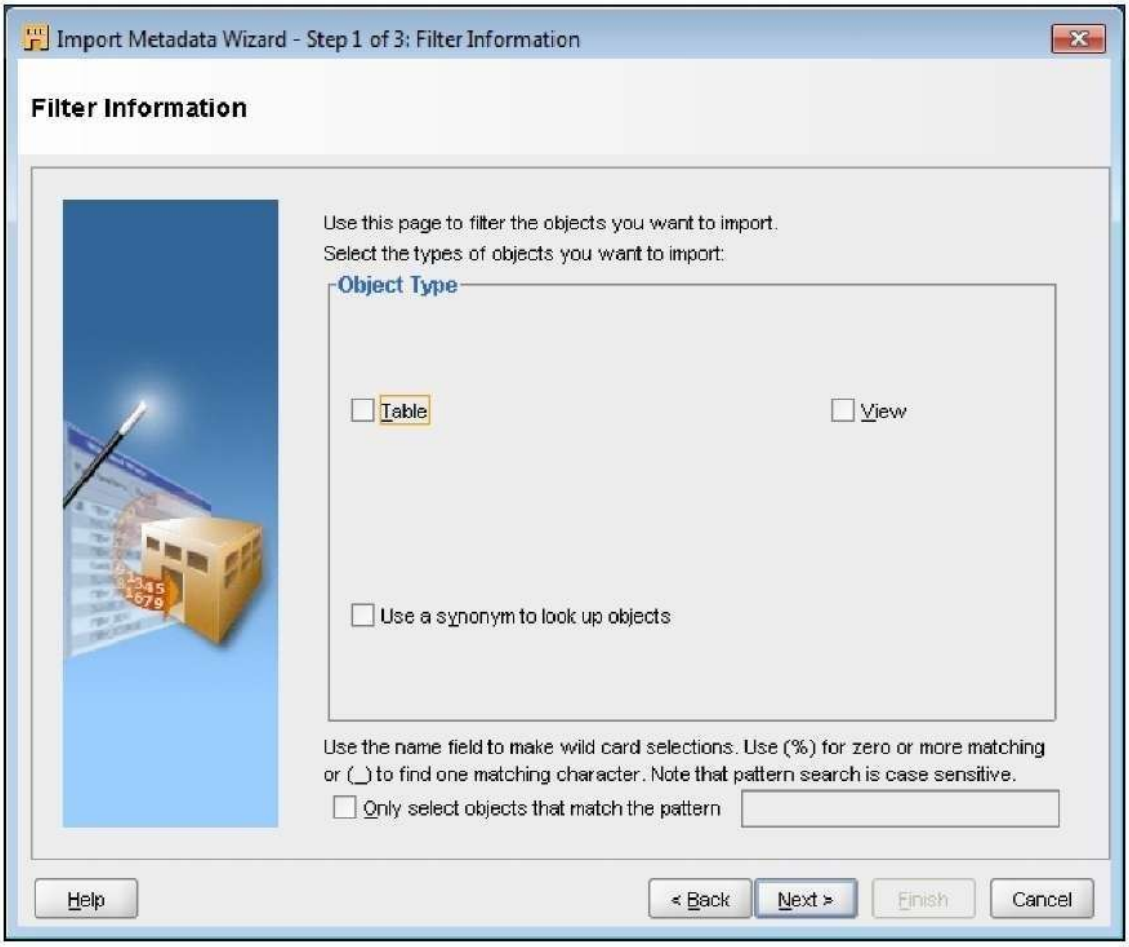
1. We are going to begin by right-clicking on the ACME_WS_ORDERS module name under the Databases | Oracle node in the sProjects tab and selecting Import and then Database Objects… from the pop-up menu.
The Import submenu that appears when we right-click and choose Import on a database node has more than one choice. We might be tempted here to select the first choice, Warehouse Builder Metadata. for this import; however that is for a different kind of import, Metadata exports and imports that we'll look at in topic 9. In this new release, that option has been added to the context menus on the various objects and to differentiate that process from the task of importing data objects from source database systems, the option we want has been named Database Objects..

We will then be presented with the Import Metadata Wizard. This is the same wizard that will be used for importing from any of the available source data options, databases, files, and so on. It will tailor its prompts for the particular type of source we selected.

We may at this point see a pop-up warning about the connection not being set and to click OK to set connection details. That's frequently just because the password didn't get saved with the connection. It will display the connection details where we can fill in the proper password before continuing.
2. Click on the Next button on the Welcome screen and we'll be presented with a screen labeled Step 1 of the process where we choose what to import. The following image is what it looks like for an Oracle database:

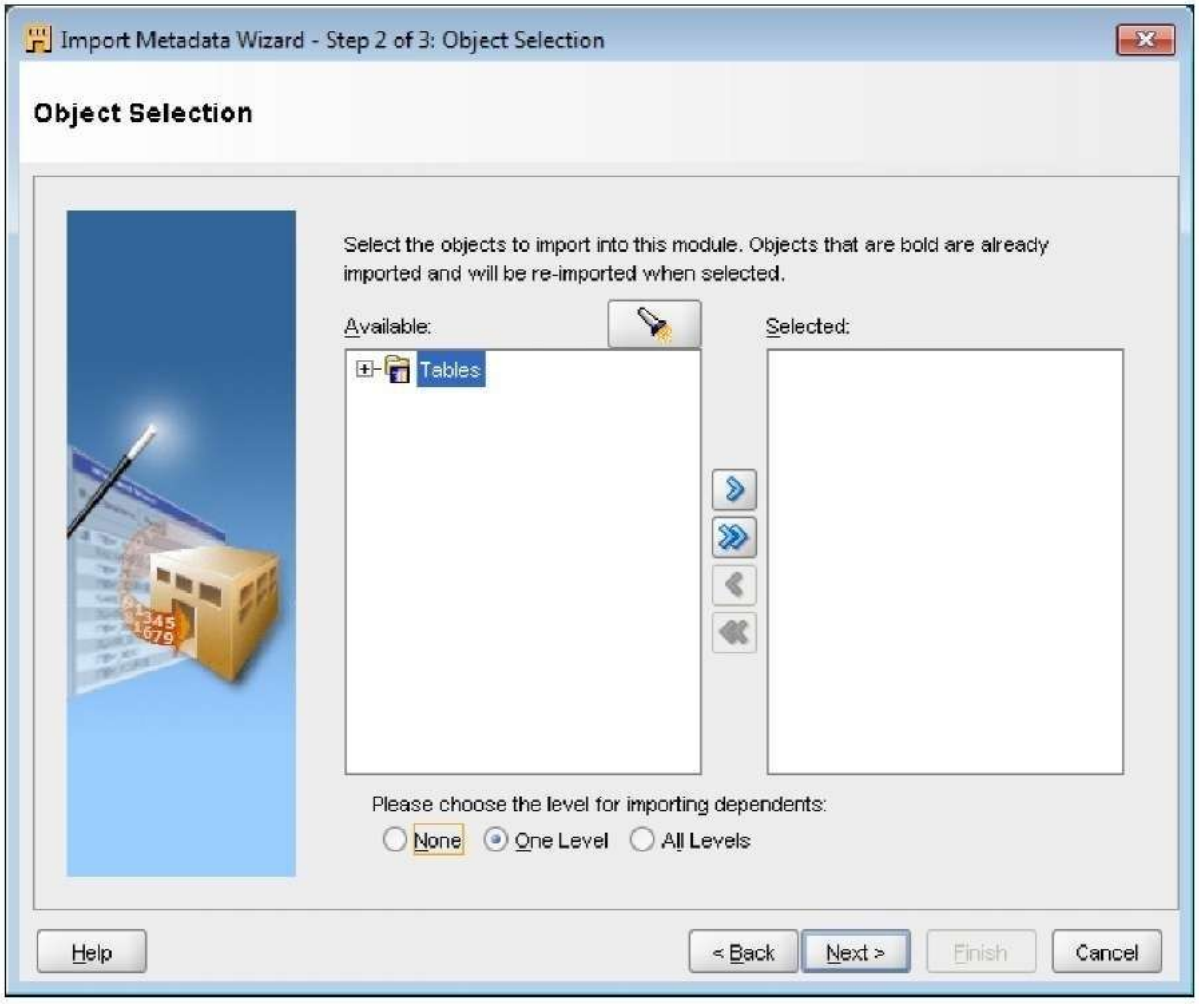


We can make selections on this screen to filter out just what we want to import, or we can leave everything checked to be able to import anything. This screen will appear slightly different depending on what type of source we're importing from. We will have all these options for an Oracle database, but for our ODBC connection to the SQL Server database, it will have checkboxes for just Table and View as shown next:



There will also be a checkbox for whether to use synonyms to look up the objects, and a text box where we can enter a search pattern to use if we want to further refine what is available to us. We're just going to check the Table checkbox since it is only tables we'll be importing from either source database. Checking the Use a Synonyms box means that if there are any synonyms defined (alternative names for database objects), then the import wizard will use those names and present them to us; otherwise it uses the underlying actual object names. As there are no synonyms being used in the ACME_WS_ORDERS or ACME_POS source databases, it will not make a difference whether it's checked or not.

3. Click on Next to move on to Step 2:



This screen is where we will choose the specific objects that we wish to import. There will be an entry in the left window for each of the boxes we left checked in Step 1. Notice (at the bottom) the buttons for choosing the level for importing dependents. The Import Wizard can automatically import other objects that might depend on the object we're selecting based on foreign key definitions that it detects in the source database. The number of levels means how far it goes in tracing foreign key relationships. If we say one level, which is the default, then it will import any tables that have foreign key relationships to the table selected but will not check those tables for relationships. If you say All Levels, then it will follow relationships until it doesn't find any further relationships. We're going to select all the tables in our acme_ws_orders schema or the acme_posSql Server schema to import, so this setting will not have an effect on what gets imported. Therefore, we'll leave it set to the default.

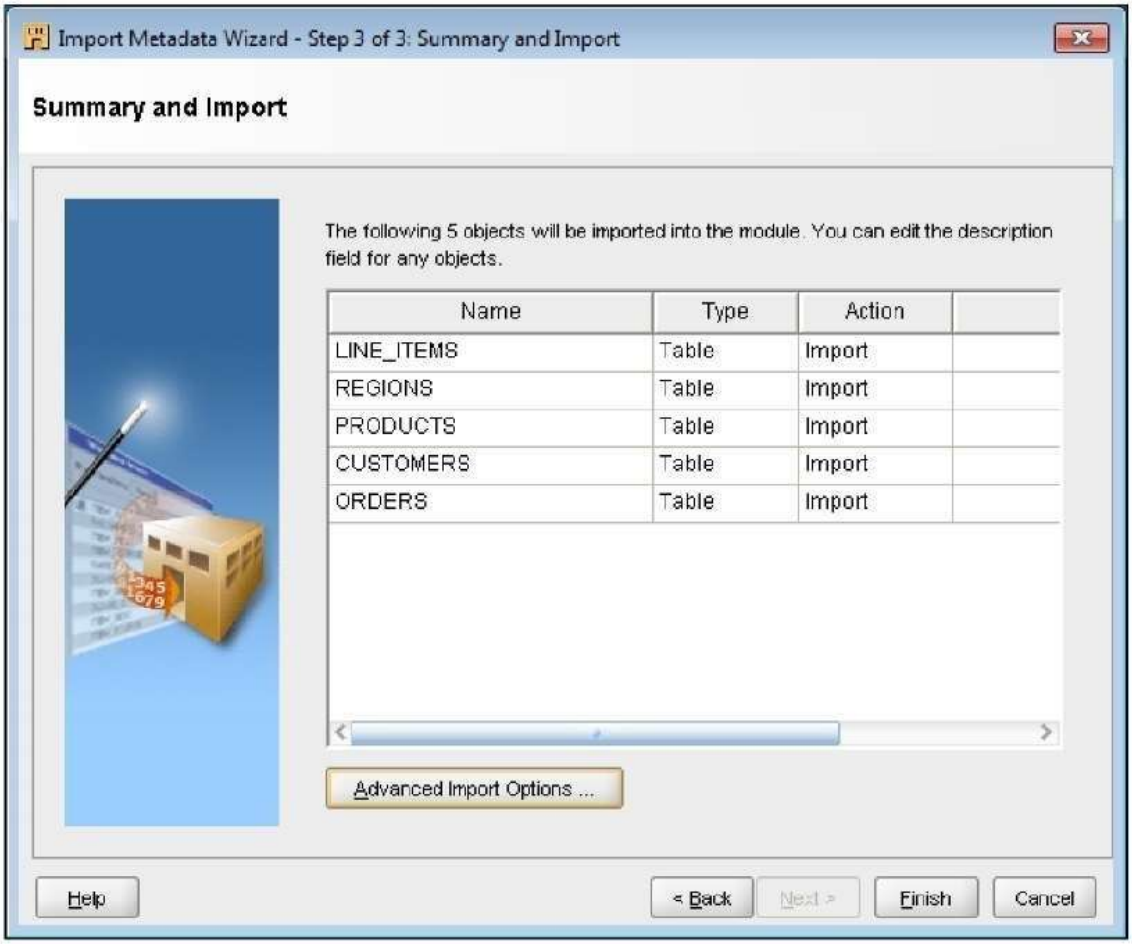
4. Click on the plus sign beside the Tables entry to see the complete list of tables to choose from. We will see all of the website orders' database tables that we discussed earlier and if importing the ACME_POS Sql Server database, all of the source point of sale transaction database tables.

If we were importing from a SQL Server database and had not enclosed the schema name in double quotes and made it lowercase, we would not see any tables show up here. There would be no error message at all. It would look like it was doing something and then the plus sign would change to a minus sign as if it had expanded and displayed all the tables but nothing would show up. In fact, no error is generated as far as the Warehouse Builder is concerned. It submits a SQL statement to the source database requesting a list of table names where the owner is the schema name we included in the location but it makes that schema name uppercase unless we use the double quotes. The schema name in MS SQL Server is lower case and therefore the SQL statement returns no results and so nothing displays. So, when setting up a SQL Server location, be sure to make the schema name lowercase and use double quotes around it.

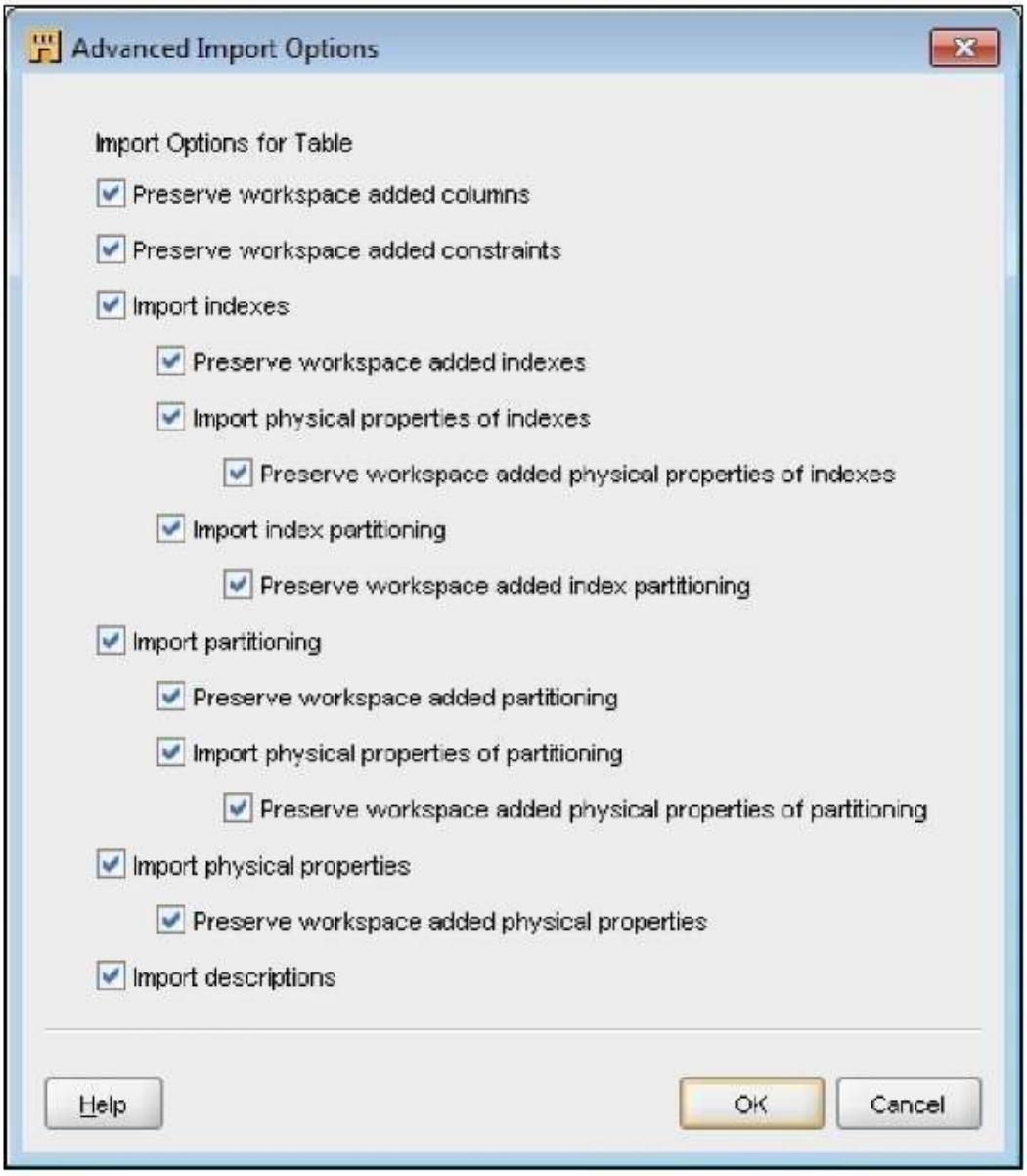
5. Clicking on the plus sign to expand an entry on this page is the first time the wizard will actually make a connection to the source database. If we saved the connection information before and didn't test it to make sure it worked, this is where we'll find out.

The table names will display under the tables entry. If we've already imported any of the tables previously, those will be displayed in bold. We can re-import them to pick up any changes that might have been made to them. We're going to click on all the tables, and then click on the single right arrow (>) in the middle of the screen to move those tables over to the right side. This will signify that we want to import them. As we want all the tables this time, we could alternatively click on the Tables entry itself and then on the single right arrow (>) to move that entry and everything in it over to the right or just click the double right arrow (>>) to move everything. At this point, if we had one of the options checked for importing dependents and had not already selected the dependents to import, it would display a dialog box. This dialog box would inform us of any additional objects it detected as dependents that it was going to automatically add for us.

6. We'll click on the Next button to proceed to the Summary and Import page where it will summarize the selections we've made and tell us the action it is going to take for each selection—whether to create or re-import the object. There is also an Advanced Import Options… button that will be available on that screen as we can see in the following image:



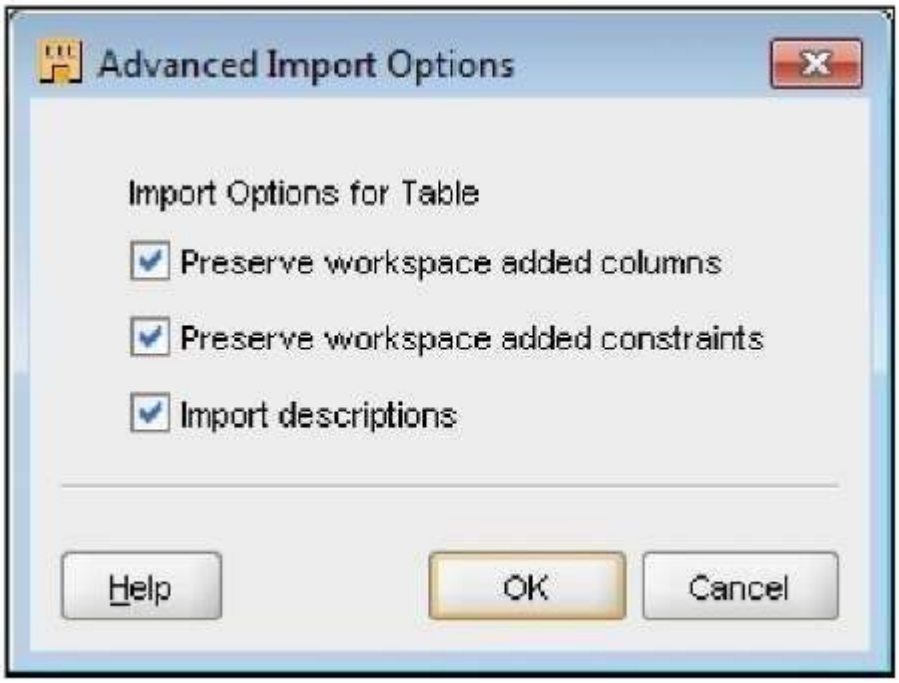
Clicking on the Advanced Import Options... button presents us with a dialog box similar to the following screenshot:



This dialog box will be slightly different, depending on the type of object and the type of source being imported. The screenshot we just saw is an example for a table from an Oracle database. It specifies whether to import certain features, such as indexes or physical properties, and also whether any possible changes we've made to the objects in the Warehouse Builder workspace after import should be preserved.

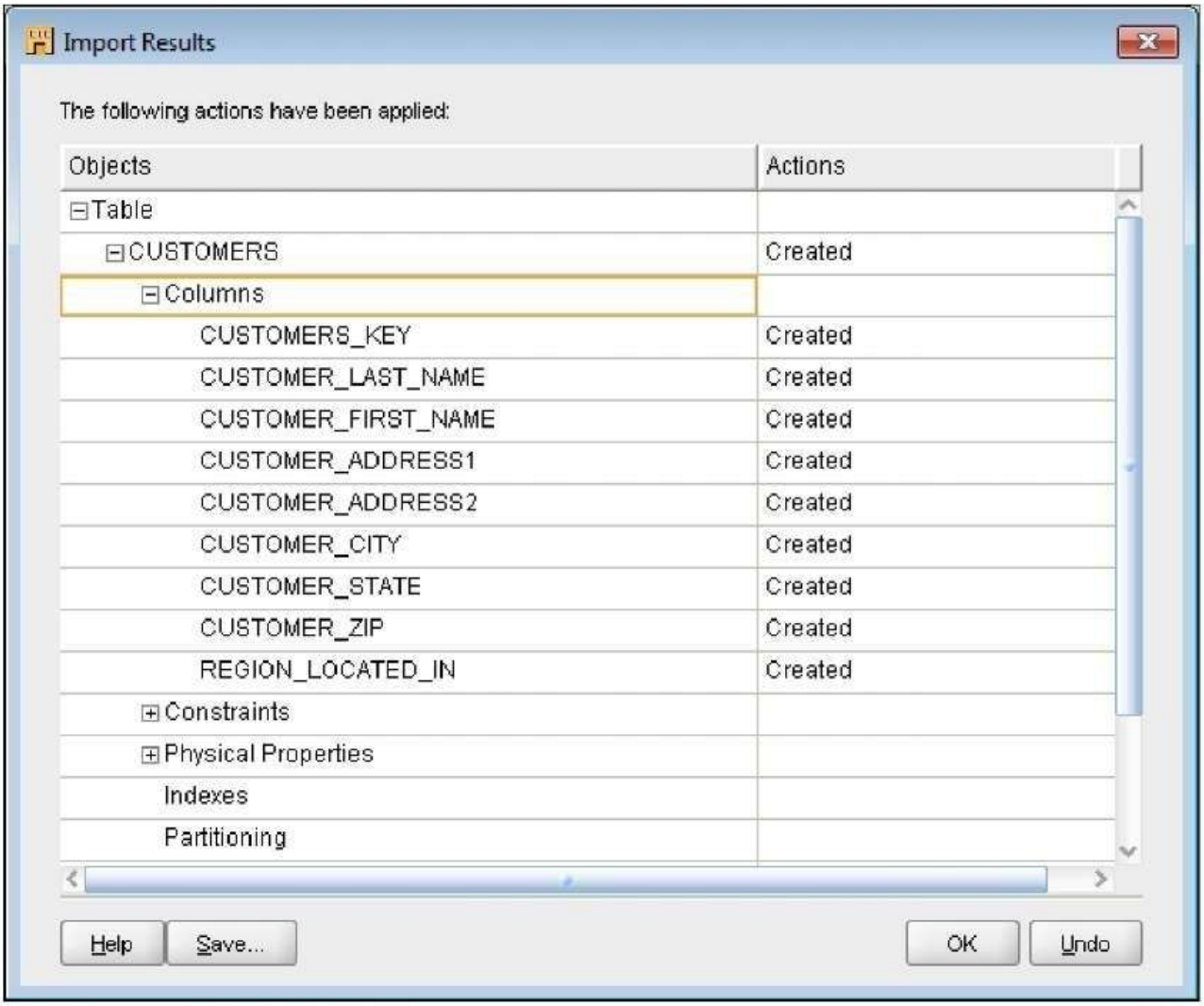
The option for preserving changes made in the workspace would definitely be something we will need to consider if we subsequently import objects from SQL Server after defining them manually. If we don't uncheck the boxes for preserving workspace changes and we've already created all the tables manually as we're about to do, it will leave all our column definitions in place and add new columns for each of the columns in the table. In that case, we would definitely want to uncheck the preserve checkboxes so our manual edits are replaced with a clean copy.

The following is what we'll see for Advanced Options when importing from a SQL Server database using the ODBC module:



We have verified on the Summary and Import screen that we have included everything we want to import and we don't need to bother with unchecking any of the advanced import options. So we will click on the Finish button, which will begin the import process.

7. During the import, a status dialog box will be displayed showing the progress of the import as each object is imported. When it completes, we'll be presented with the final Import Results screen showing the status of the import. We can click on the plus sign beside each entry to see the details as shown in the following screenshot with the Customers table expanded to show each of the columns:



The other buttons you have on this screen are:

° A Save button which will allow us to save a Metadata Import Result Report log file so we can have a record of the results of our import if desired.

° An Undo button that we could click on at this point to cancel the import. The Import Metadata Wizard has not actually saved any information to the database yet, so clicking on the Undo button will just throw away what we've done so far and not make any changes to the database.

° The OK button will save the changes to the module in Projects tab from which we performed the import.

In this case, clicking on OK is going to save the imported tables in the ACME_WS_ORDERS module that we created under the Databases | Oracle node or the ACME_POS module under the Databases |

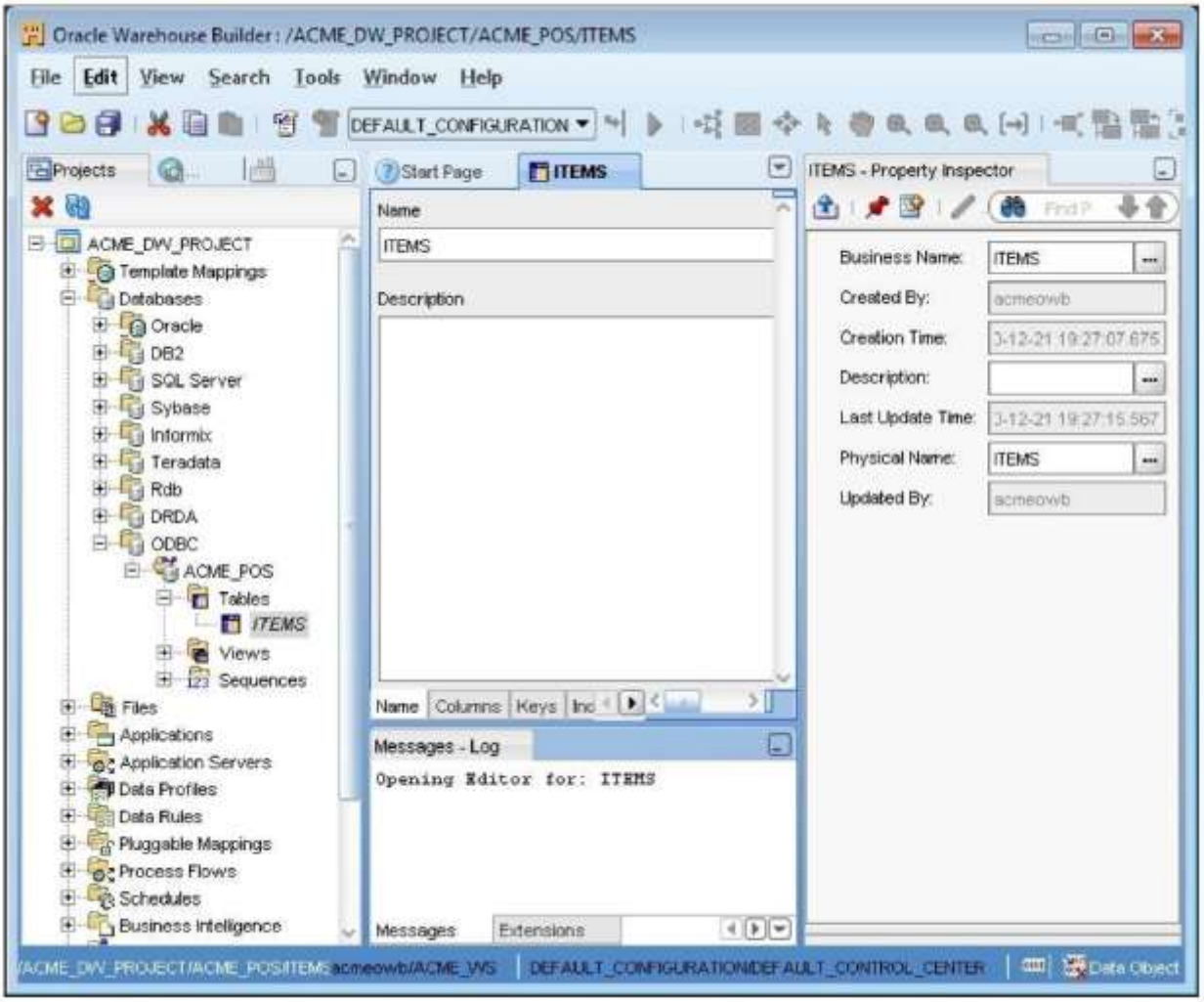
ODBC node. We can verify this by going back to the Projects tab window and expanding the appropriate module if it's not already expanded by clicking on the plus sign, and we should see the list of tables.
Congratulations! We've imported our first set of objects into the Warehouse Builder.

Let's take a look at the Table Editor (a data object editor for editing tables) now as another alternative available to us for entering the metadata for our source database tables. If the import were to fail for some reason, we could always fall back to manually entering the information for our source databases. A third option would be to copy and paste table metadata across modules. We'll discuss copying and pasting metadata in topics 9 and 10.
We should save our work at this point. So we'll select Design | Save All from the toolbar menu of the Design Center application or press the Ctrl + S key combination to save our work.

Defining source metadata manually with the Table Editor

Before we can continue building our data warehouse, we must have all our source table metadata created. If the automatic import via the Metadata Import Wizard were to fail for whatever reason, we must create the source metadata manually in that case. It is not a particularly difficult task. However, attention to detail is important to make sure what we manually define in the Warehouse Builder actually matches the source tables we're defining. Warehouse Builder provides contextual data object editors for creating and editing source metadata. The Table Editor is a tool we can use to create database tables in the Warehouse Builder. The steps to manually define the source metadata using the Table Editor are:

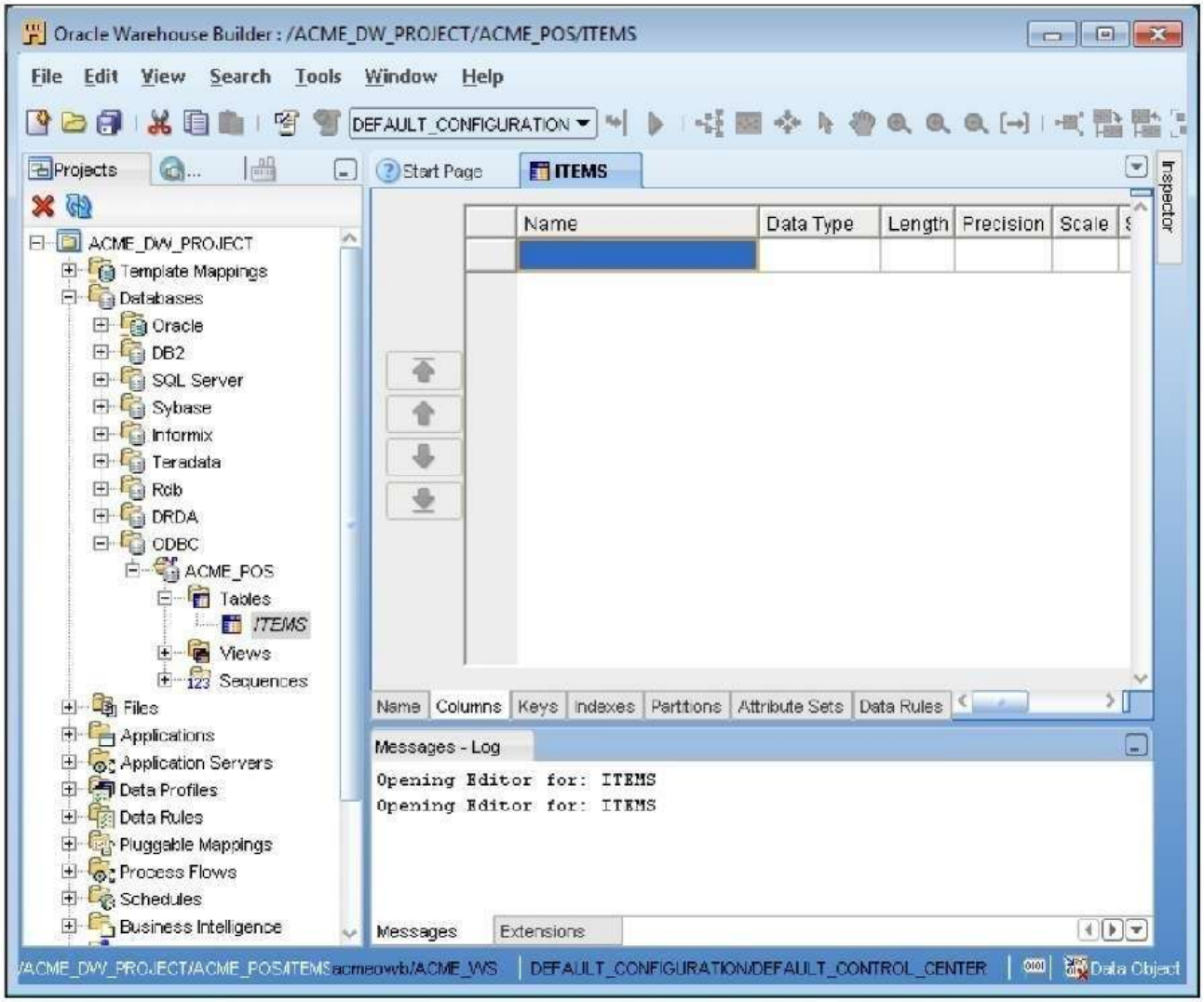
1. To start building our source tables for the POS transactional SQL Server database, let's launch the OWB Design Center if it's not already running. Expand the ACME_DW_PROJECT node and take a look at where we're going to create these new tables. We created our ACME_POS module for the SQL Server source database under the Databases | ODBC node so that is where we'll create the tables. Navigate to the Databases | ODBC node, and then select the ACME_POS module under this node. We will create our source tables under the Tables node, so let's right-click on this node and select New Table from the pop-up menu. As no wizard is available for creating a table, we are using the Data Object Editor to do this.
2. The first screen we'll be presented with is a small popup asking us to fill in the name and a description for the new table we're creating. We're going to create the metadata for the ITEMS table so let's change the name to ITEMS and click OK to continue.
3. Upon selecting OK, we are presented with the Table Editor screen on the right hand side of the main Design Center interface. It's a clean slate that we get to fill in, and will look similar to the following screenshot:



There are a number of facets to this interface but we will cover just what we need now in order to create our source tables. Later on, we'll get a chance to explore some of the other aspects of this interface for viewing and editing a data object. The interface is completely customizable also. The Sub-windows can be dragged to any position we want. For our purposes now we're going to leave the interface as it is and be working in the main Table Editor window labeled ITEMS. The fields to be edited in this Table Editor are as follows:

° The first tab it presents to us is the Name tab. We can see all the various tabs arrayed along the bottom of the ITEMS editor window. The Name is already filled in for us from the initial popup so we'll move on to define the columns.

° Let's click on the Columns tab next and enter the information that describes the columns of the Items table. To make the window easier to work with we'll minimize the Properties Inspector to make more room for the main ITEMS editor window. Just click the minimize button in the upper right corner of the Properties Inspector and we'll see it minimize along the right hand side of the interface. Our window should now look similar to the following:



° How do we know what to fill in here? Well, that is easy because the names must all match the existing names as found in the source POS transactional SQL Server database. For sizes and types, we just have to match the SQL Server types that each field is defined as, making allowances for slight differences between SQL Server data types and the corresponding Oracle data types.

The following will be the columns, types, and sizes we'll use for the Items table based on what we found in the Items source table in the POS transaction database:

```
ITEMS_KEY number(22)
ITEM_NAME varchar2(50)
ITEM_CATEGORY varchar2(50)
  ITEM_VENDOR number(22)
  ITEM_SKU varchar2(50)
  ITEM_BRAND varchar2(50)
  ITEM_LIST_PRICE number(6,2)
  ITEM_DEPT varchar2(50)
```

We'll enter each of these column names on the Columns tab of the Data Object Editor for the Items table; and as we enter each name, it will suggest data types and sizes, which may or may not be adequate. It makes a best guess based on what we enter for the name, and may or may not relate to the source data type and size. For items_key, it suggests a number with precision 22. For item_vendor, it actually suggested a varchar2 type. We simply change it to match the items_key, as we see in the SQL Server database that both these fields are defined as type int and 22 for the precision is large enough to hold an integer from SQL Server. An integer is a four-byte number, no larger than 2,147,483,647. The other character fields in the SQL Server Items table are all of the varchar type, which is the SQL Server equivalent to a varchar2 in Oracle. So we make sure they are varchar2 with sizes that match. The item_list_price is defined with both a precision and scale because that is a decimal number in the Items table in SQL Server with that precision and scale.

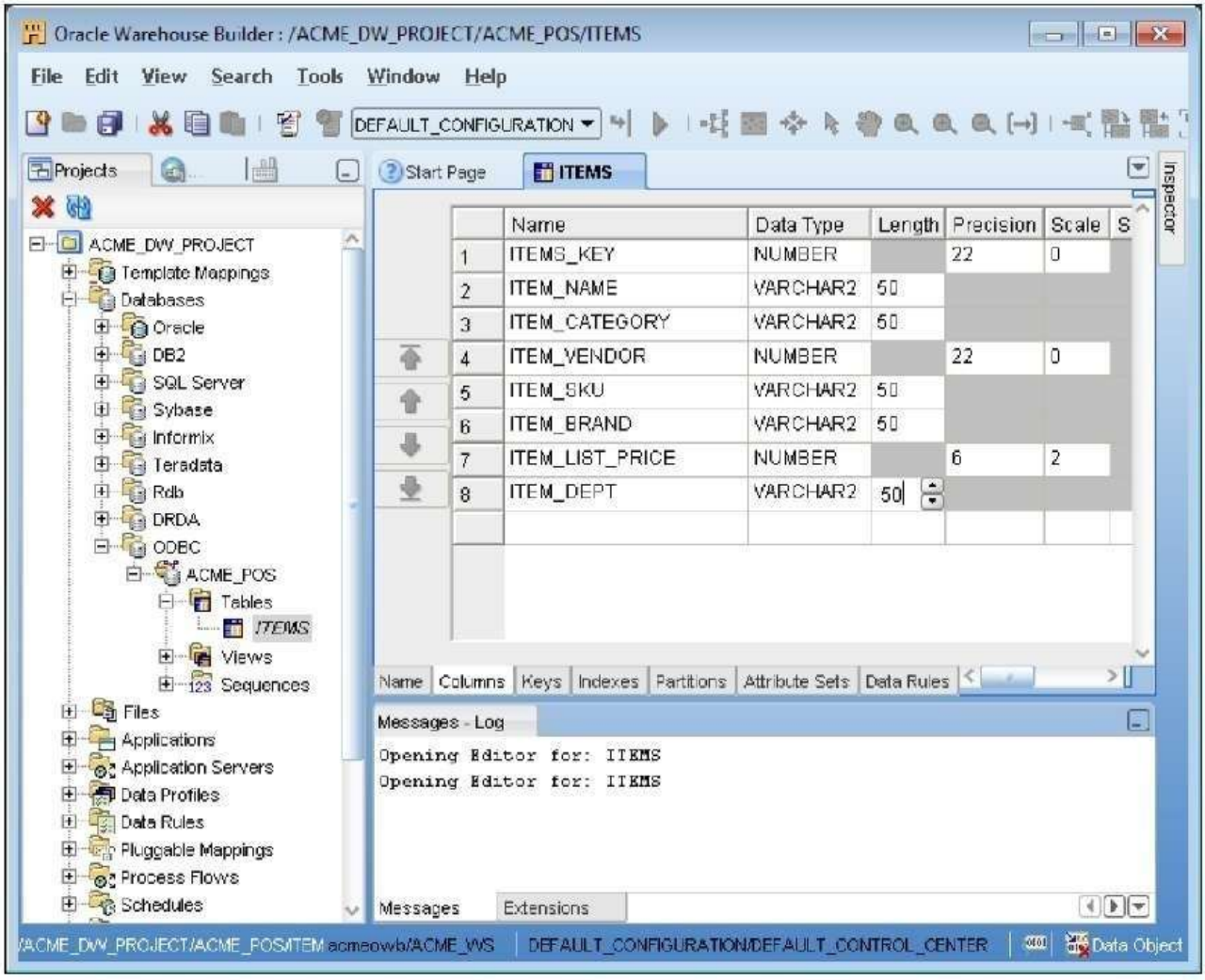
Precision and scale of numbers

Properties of number data types can include a precision and scale. Oracle allows a number data type to be specified without indicating a specific precision and scale. Precision indicates the maximum number of digits the number can contain, and scale indicates the number of decimal places to the right of the decimal. If we don't specify them, Oracle Database will accept a number of any precision and scale as long as the number doesn't fall outside the range allowed, which is between 1.0 x 10-130 and 1.0 x 10126. We would specify a precision and scale to enforce greater data integrity in the database. For example, if we enter a number that has more digits than the specified precision, it will generate an error even though the number might still fall in the acceptable range.

We don't have to worry about specifying information for any of the other tabs for this source table. The important details are the column names, and their types and sizes. Later in the topic, we'll have a chance to revisit this editor again and discuss the remainder of the tabs.

4. We can save our work at this point and close the Table Editor window now before proceeding. So we'll select File | Save All from the main menu of the Design Center, or press the Ctrl + S key combination to save our work. We can close the Table Editor window for the ITEMS table by hovering the mouse over the ITEMS tab at the top of the window and then clicking on the X that appears or by selecting File | Close from the Design Center. Closing the editor window is really a matter of preference. With this new release of the Warehouse Builder, all the editors are integrated into the main interface and as new ones are opened, new tabs will be added to the canvas on the right. We can close the Start Page also just to clean up the work area and it will be blank until we open another tool or editor.

When completed, our column list should look like the following screenshot:



We now have the option to continue this process to define the metadata for the remaining SQL Server tables that we'll need or to just do the import using the Import Metadata Wizard. If continuing to enter tables manually, the process is identical; just change the names of the tables and the types and sizes of the columns to match their respective tables. We will not need all the tables defined in the ACME_POS database—only a subset is used throughout the remainder of the topic to build the

data warehouse. The tables needed are the pos_transactions, registers, stores, and regions tables. The column information for each of them is provided here for reference and help in creating the corresponding tables in the Warehouse Builder:

POS_TRANSACTIONS

POS_TRANS_KEY number(22)
SALES_QUANTITY number(22)
SALES_ASSOCIATE number(22)
REGISTER number(22)
ITEM_SOLD number(22)
DATE_SOLD date
AMOUNT number(10,2)

REGISTERS

REGISTERS_KEY number(22)
REGISTER_MANUFACTURER varchar2(60)
MODEL varchar2(50)
LOCATION number(22)
SERIAL_NO varchar2(50)

STORES

STORES_KEY number(22)
STORE_NAME varchar2(50)
STORE_ADDRESS1 varchar2(60)
STORE_ADDRESS2 varchar2(60)
STORE_CITY varchar2(50)
STORE_STATE varchar2(50)
STORE_ZIP varchar2(50)
REGION_LOCATED_IN number(22)
STORE_NUMBER varchar2(10)

REGIONS

REGIONS_KEY number(22)
REGION_NAME varchar2(50)
CONTINENT varchar2(50)
COUNTRY varchar2(50)