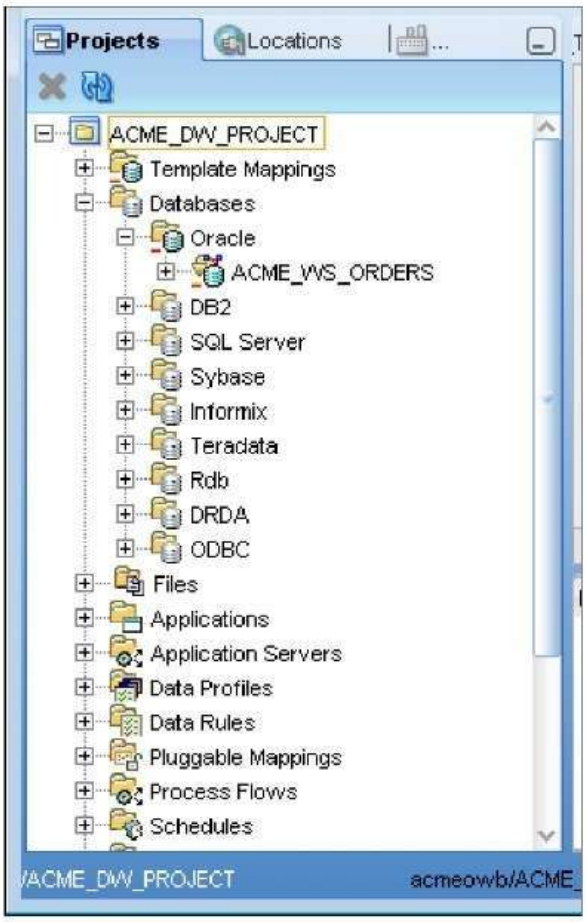
The Warehouse Builder contains a number of objects, which we can use in designing our data warehouse, that are either relational or dimensional. OWB currently supports designing a target schema only in an Oracle database, and so we will find the objects all under the Oracle node in the Projects tab. Let's launch Design Center now and have a look at it. But before we can see any objects, we have to have an Oracle module defined to contain the objects. If you've been following along and working through the examples in this topic, so far you should have one module already defined for the ACME website orders database—acme_ws_orders. We created this in the last topic when we imported our metadata from that source. If that is the case, our Projects tab window will look similar to the following:



Creating a target user and module

We need a different module to create our target objects in. So before going any further, let's create a new module in the Projects tab for our target to hold our data warehouse design objects. However, before we can do that, we should have a target schema defined in the database that will hold our target objects when we deploy them.

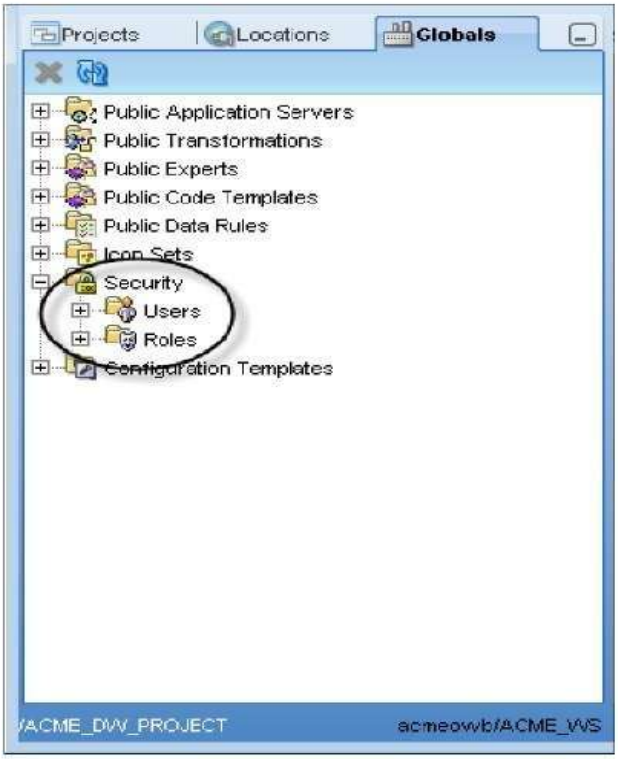
So far we have discussed many different components such as the repository, workspaces, the design center, and so on. So, it can be confusing to know exactly where our main data warehouse is going to be located. The target schema is going to be the main location for the data warehouse. When we talk about our "data warehouse" after we have it all constructed and implemented, the target schema is what we will be referring to. Amid all these different components we discussed that compose the Warehouse Builder, the target is where the actual data warehouse will be built. Our design will be implemented there, and the code will be deployed to that schema by OWB to load the target structure with data from the sources.

Every target module must be mapped to a target user schema. Back in topic 1, when we ran the Repository Assistant to create the repository and workspace, we created the acmeowb user as the repository owner and mentioned that this user can be a deployment target for our data warehouse. However, it does not have to be the target user. It's a good idea to create a separate user schema to become the target so that user roles in our database can be kept separate. Using the OWB repository owner schema would mean our target data warehouse would have to be on the same database server as our repository. In large installations, that will most likely not be the case. So for maximum flexibility, we're going to create a separate user schema. In our case, that user will be created in the same database as the repository; but it can be moved to another database easily if we expand and add more servers.

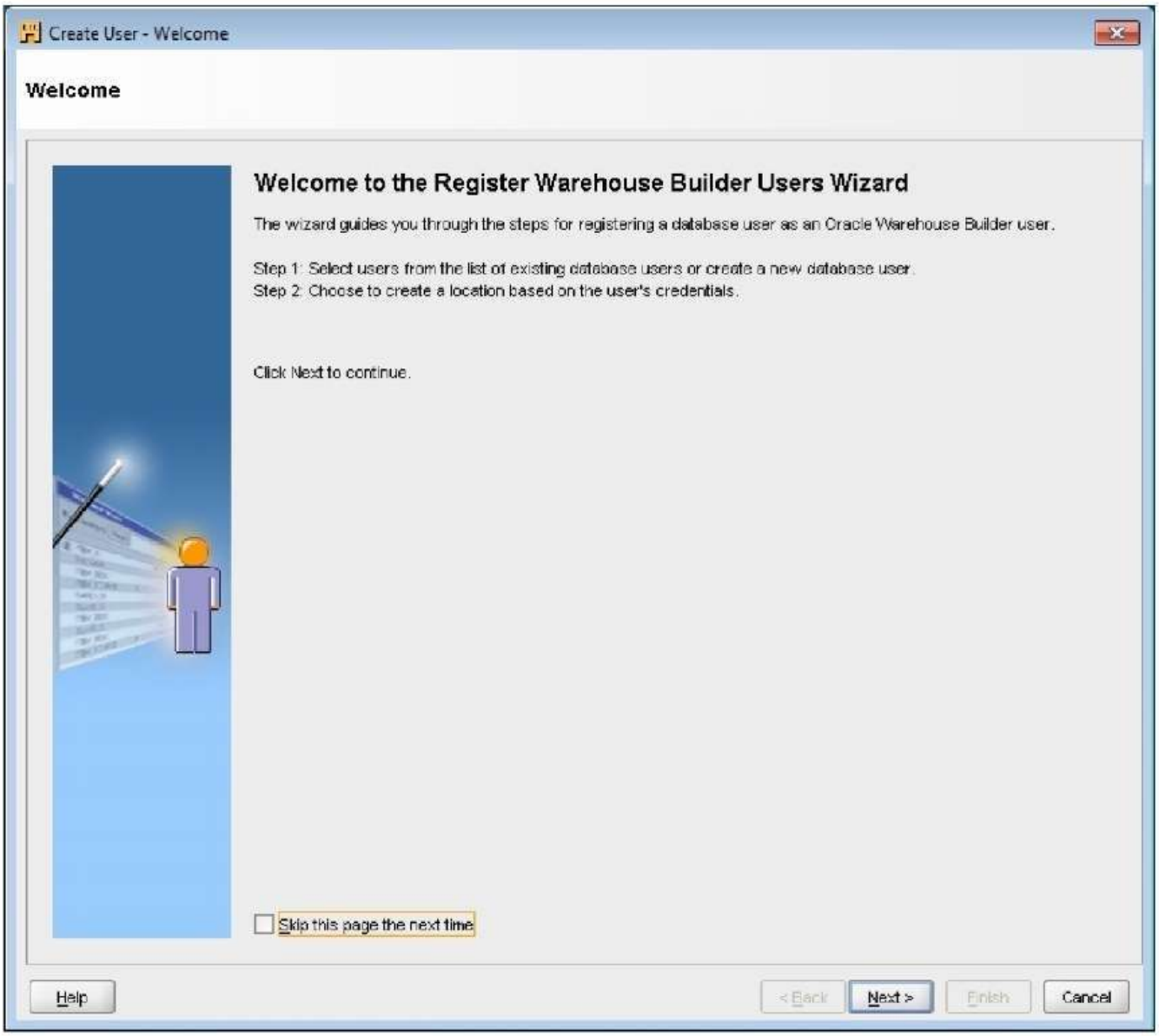
Creating a target user

There are a couple of ways we can go about creating our target user—create the user directly in the database and then add to OWB, or use OWB to physically create the user. If we have to create a new user, and if it's on the same database as our repository and workspaces, it's a good idea to use OWB to create the user, especially if we are not that familiar with the SQL command to create a user. However, if our target schema were to be in another database on another server, we would have to create the user there. It's a simple matter of adding that user to OWB as a target, which we'll see in a moment. Let's begin in the Design Center under the Globals tab. We talked about that Globals tab back in our introduction to the Design Center in topic 2. There we said it was for various objects that pertained to the workspace as a whole.

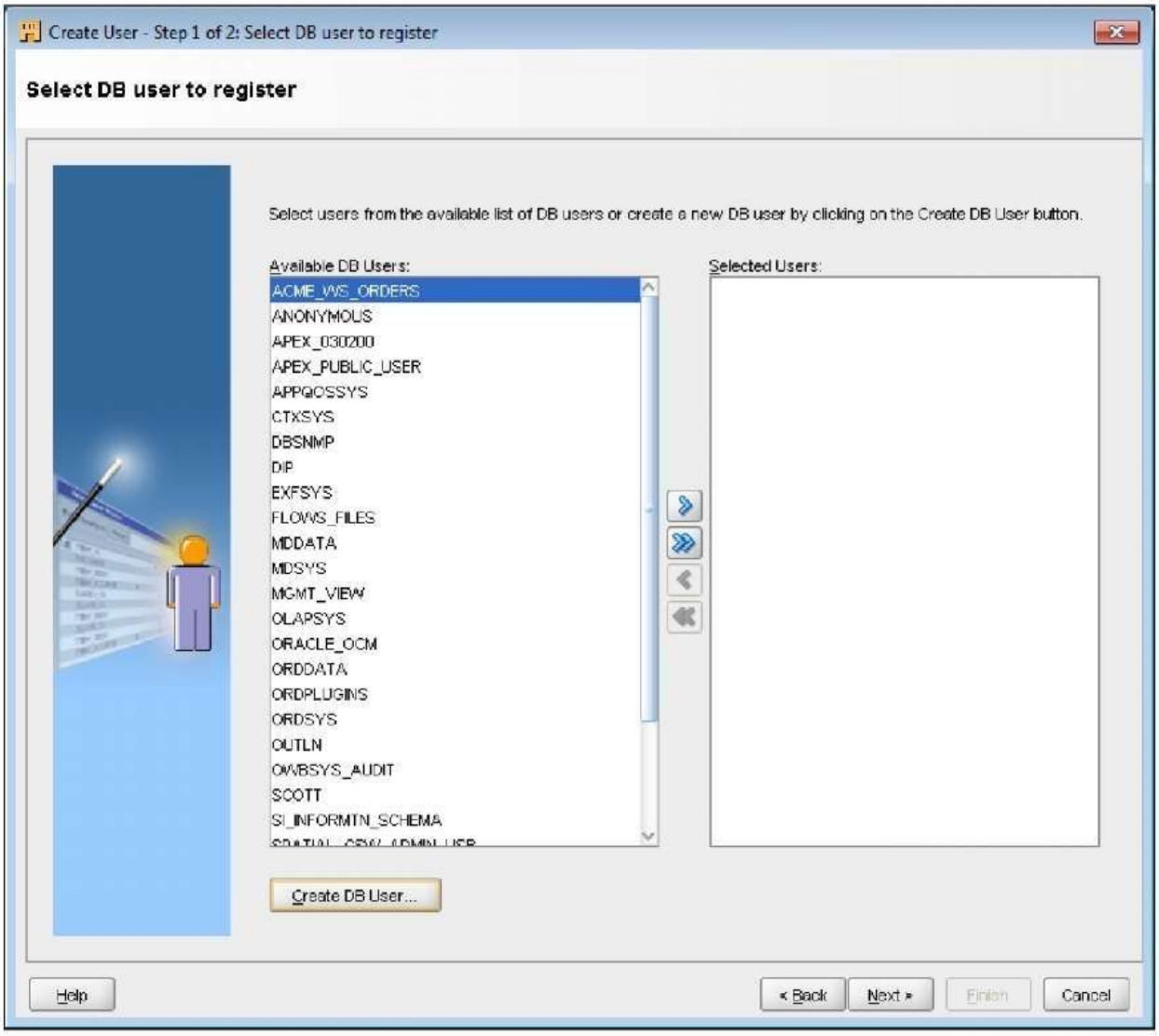
One of those object types is a Users object that exists under the Security node as shown here:



Right-click on the Users node and select New User... to launch the Create User dialog box as shown here:

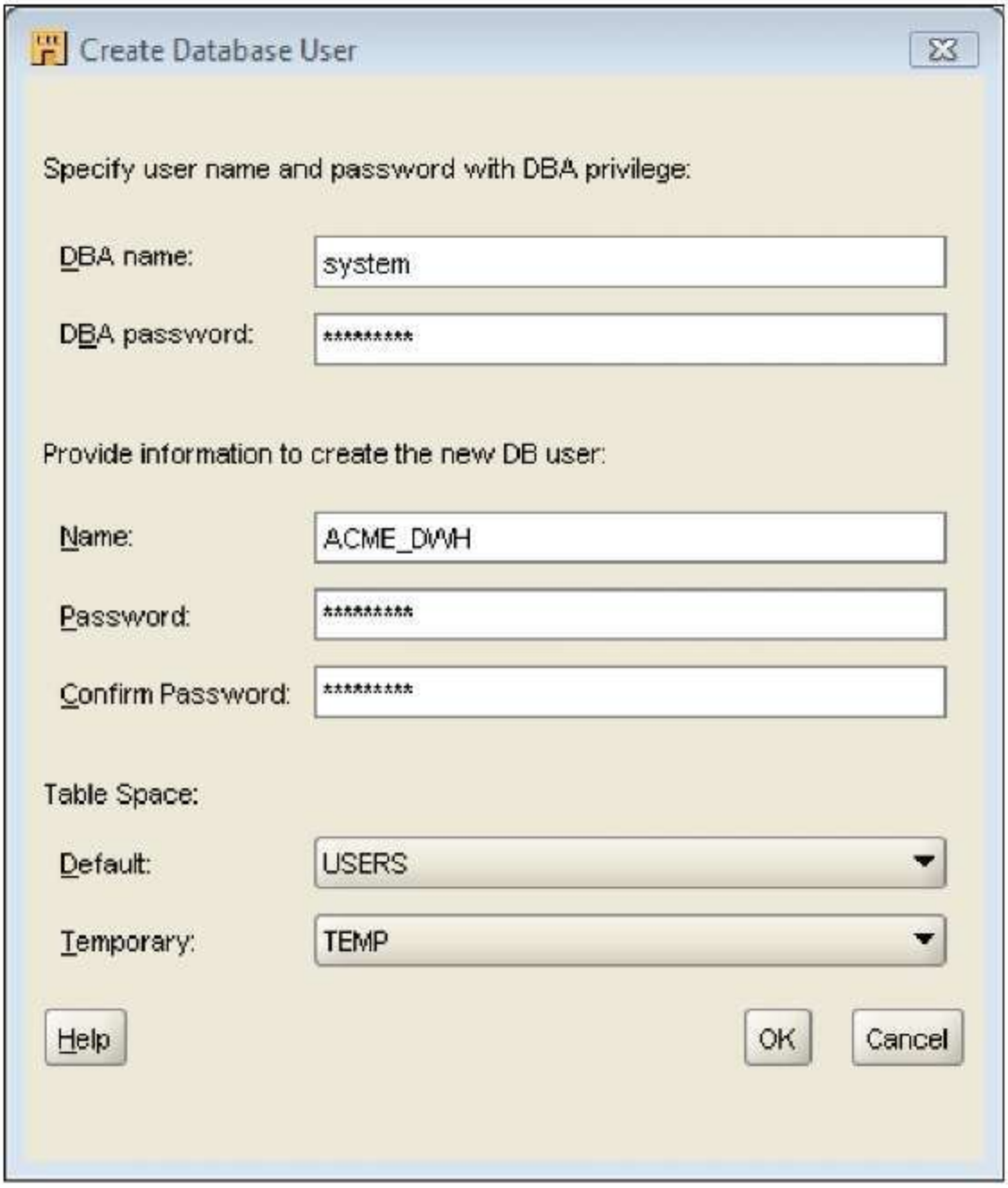


With this wizard, we are creating a workspace user. We create a workspace user by selecting a database user that already exists or create a new one in the database. We'll just click the Next button to move on to step 1 as shown next:

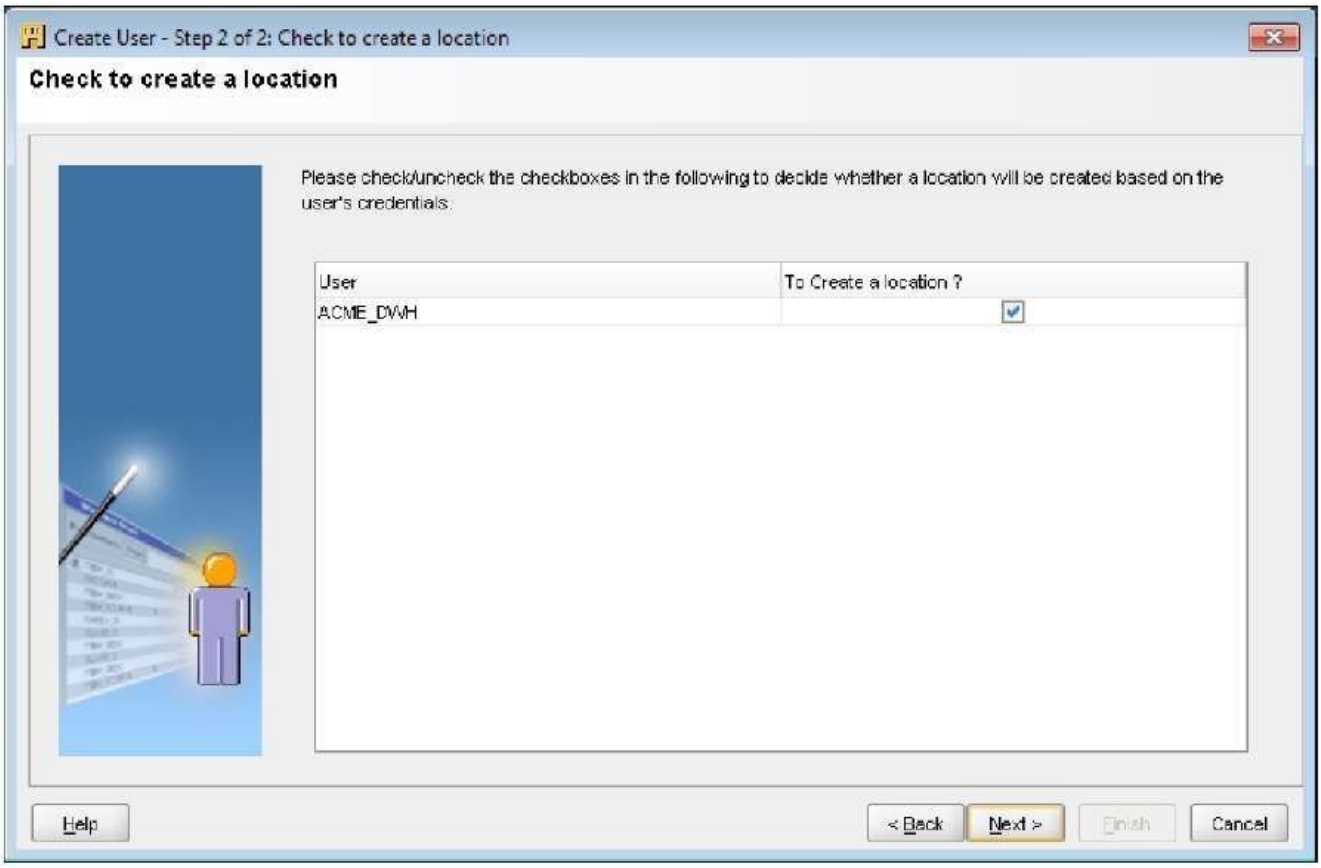


If we already had a target user created in the database, this is where we would select it. We're going to click on the Create DB User... button to create a new database user.

We need to enter the system username and password as we need a user with DBA privileges in the database to be able to create a database user. We then enter a username and password for our new user. As we like to keep things basic, we'll call our new user ACME_DWH, for the ACME data warehouse. We can also specify the default and temporary tablespace for our new user, which we'll leave at the defaults. The dialog will appear like the following when completely filled in:



The new user will be created when you click on the OK button, and will appear in the right hand window of the Create User dialog already selected for us. Click on the Next button and we'll be presented with the second step of the user creation process, whether to create a location using the user credentials or not as shown in the following image:



We discussed locations in the last topic and saw how they were required for the Warehouse Builder to know where to connect to for the various tables and other database objects defined in modules we've defined in our project. Since we're going to use this new user we've just created as an eventual target for creating our data warehouse in then we will need to leave this checkbox checked so it creates a location based on this user. We could be just creating another authorized database user for accessing the workspace but not intending to use it as a target for any object creation in which case we wouldn't need a location defined for it. We'll leave the check box checked and click the Next button to proceed.

The final screen is just the Summary screen indicating the user to be created and whether a location will be created or not. We'll just click the Finish button and the user will be registered with the workspace, and we'll see the new username if we expand the Users node under Security in the Globals tab. Since we had indicated that we wanted a location created also, a location for the user will be evident on the Locations tab under the Locations…Databases…Oracle node. We can continue with creating our target module now that we have a user defined in the database to map to.

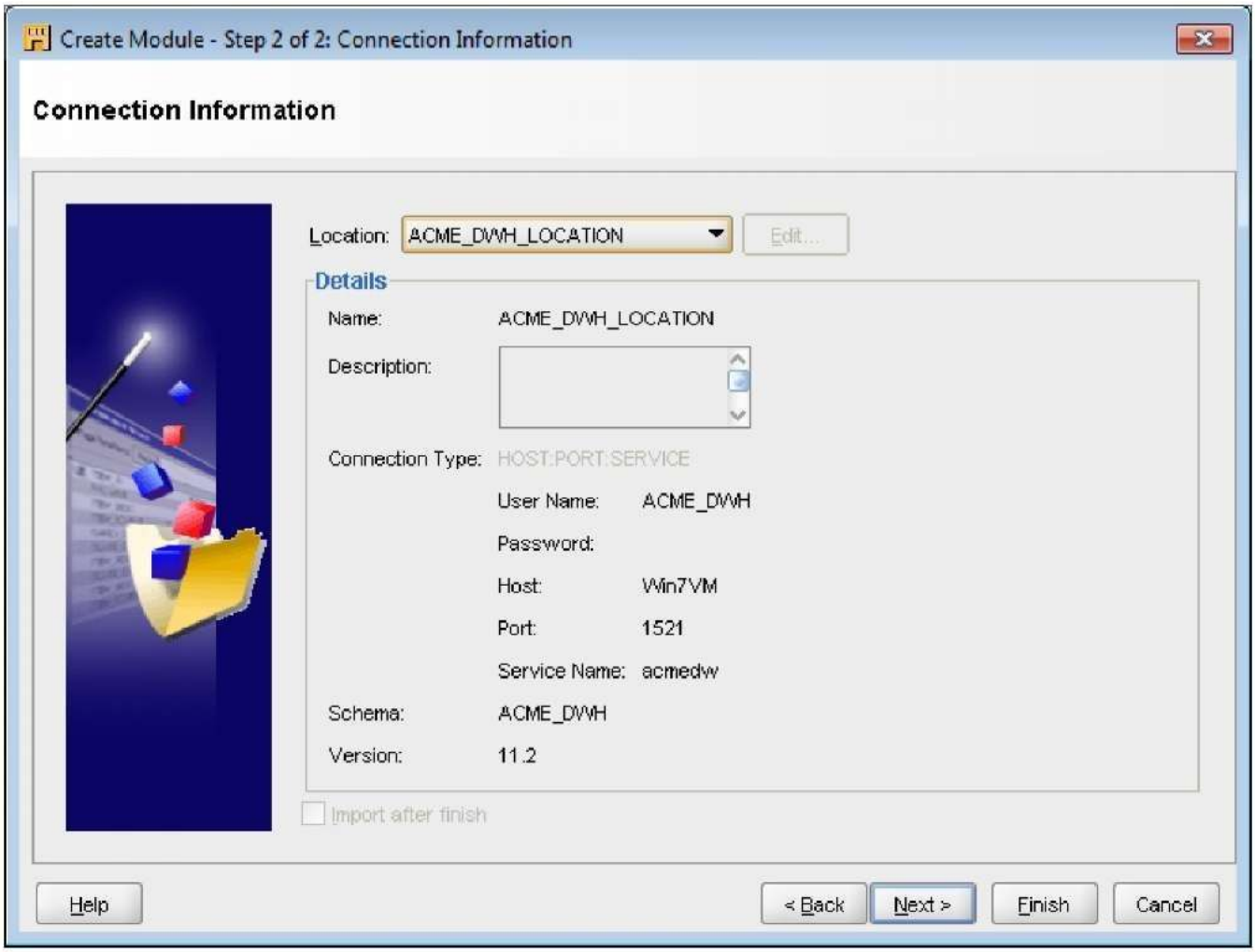
Notice that we could indicate whether we wanted a location created or not but had no way to specify the database location information. This is because it creates the user on the local database we were connected to when we logged into the Design Center, which is the location of our repository and workspaces. Due to this, this method can only be used to create the user if it is on the local database.

In the next section where we create our target module, we'll get to specify the location and that dialog box will allow us to specify a remote database if needed.

Create a target module

We'll follow the same steps as we did in the last topic where we created the ACME_ ws_orders module. Right-click on the Oracle object under Databases and select New Oracle Module… from the pop-up menu to launch the Create Module Wizard and step through the process. We'll name this module ACME_DWH for ACME Data Warehouse.

The next step is for creating or selecting a location to use. Since we just created the user to use as the target user and had the Warehouse Builder create the location automatically for us there is a location available now on the local server we can use. We'll just click the drop down and select the location labeled ACME_DWH_ LOCATION. If we're creating our own test system, the source location may very well be the same as our target. But in real-world situations, it will likely be in a different database on a different server. If we had created a target user schema on a different database, this is the point at which we would be able to enter the connection information for that user in order to associate our target module with that user and make it a target. We would just create a new location by clicking the Edit button on the default ACME_DWH_LOCATION1 to specify the connection details for that other database.
We're not going to create a new location but will be selecting an existing one and for reference, the Step 2 screen should look like the following for selecting the location of the target module:



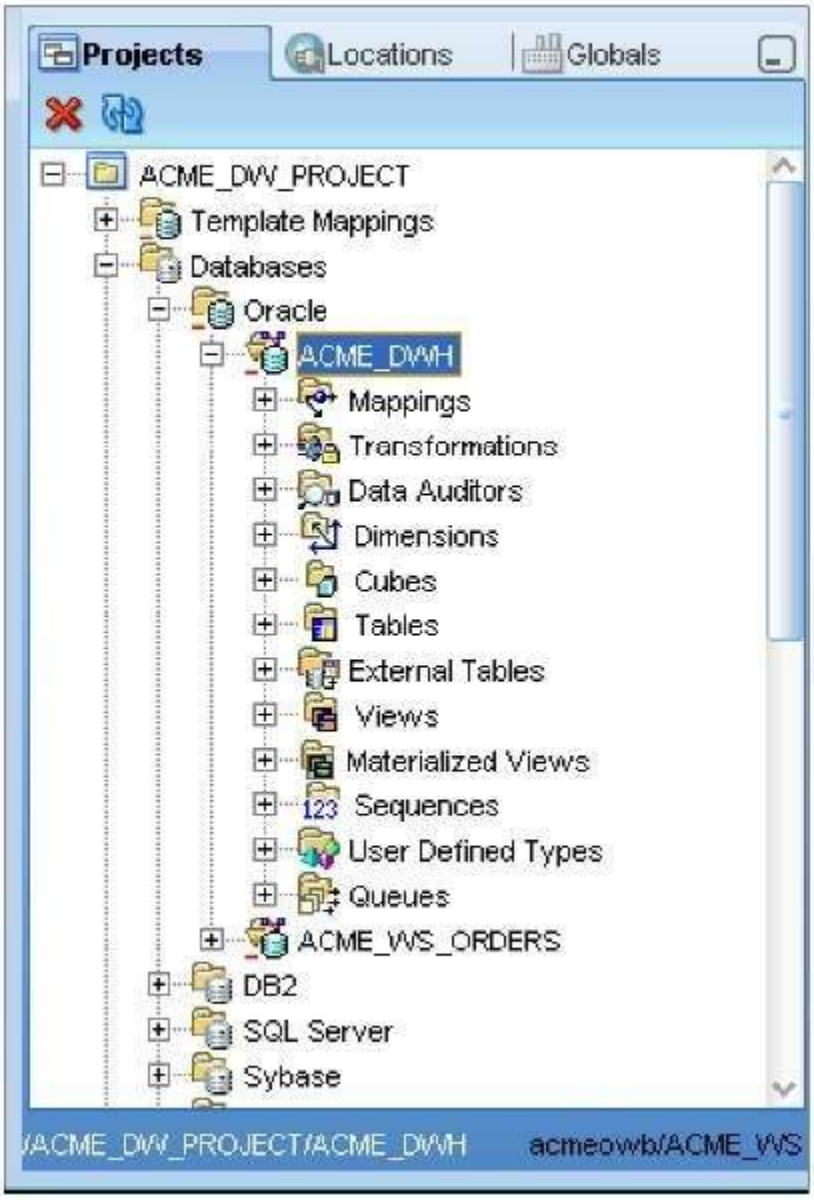
The User Name is the user we just created for this very purpose in the previous section. There is no password set for that user in the location yet but it will prompt us for that the first time we attempt to use it. The Host setting of Win7VM will be whatever the name is of the computer its running on so will vary. The Warehouse Builder uses the actual local computer name when creating the location for us rather than localhost but either will do.
If we had specified a user on a remote database the location information (Host, Port, and Service Name) would specify a user in another database if needed. If our user were not in this database, we would have just entered his or her appropriate host and port for the location and the service name of that remote database.

Now that we have our target database schema and a target module defined, which is associated with a location pointing to that target schema, we will now have two Oracle modules under our Oracle object in the Projects tab. We can continue our discussion of the design objects available to us in the

Warehouse Builder for designing our database. First, let's make sure we save our work so far by using the Ctrl+S key combination or by selecting Design | Save All from the main menu.

OWB design objects

Looking at our Projects tab window with our target Oracle module expanded, we can see a number of objects that are available to us as shown here:



There are objects that are relational such as Tables, Views, Materialized Views, and Sequences. Also, there are dimensional objects such as Cubes and Dimensions. We just discussed relational objects versus dimensional objects. We have decided to model our database dimensionally

and this will dictate the objects we create. From the standpoint of providing the best model of our business rules and representing what users want to see, the dimensional method is the way to go as we already discussed. Most data warehouse implementations we encounter will use a dimensional design. It just makes more sense for matching the business rules the users are familiar with and providing the types of information the user community will want to extract from the database.
We are thinking dimensionally in our design, but what about the underlying physical implementation? We discussed the difference between the relational and multidimensional physical implementation of a database, and now it's time to see how we will handle that here. The Warehouse Builder can help us tremendously with that because it has the ability to design the objects logically using cubes and dimensions in a dimensional design. It also has the ability to implement them physically in the underlying database as either a relational structure or a dimensional structure simply by checking a box.

In general, which option should be chosen? The relational implementation is best suited to large amounts of data that tend to change more frequently. For this reason, the relational implementation is usually chosen for the main data warehouse schema by most implementers of a data warehouse. It is much better suited to handling the large volumes of data that are imported frequently into the data warehouse. The multidimensional implementation is better suited to applications where heavy analytic processing is required, and so is a good candidate for the data marts that will be presented to users.
To be able to implement the design physically as a dimensional implementation with cubes and dimensions, we need a database that is designed specifically to support OLAP as we discussed previously. If that is not available, then the decision is made for us. In our case, when we installed the Oracle database in topic 1, we installed the Enterprise Edition with default options, and that includes the OLAP feature in the database, so we have a choice to make. Since we're installing our main data warehouse target schema, we'll choose the relational implementation.
For a relational implementation, the Warehouse Builder actually provides us two options for implementing the database: a pure relational option and the relational OLAP option. If we were to have the OLAP feature installed in our database, we could choose to still have the cubes and dimensions implemented physically in a relational format. We could have it store metadata in the database in the OLAP catalog, and so multidimensional features such as aggregations would be available to us. We could take advantage of the relational implementation of the database for handling large volumes of data, and still implement a query or reporting tool such as Oracle Discoverer or Oracle Business Intelligence Enterprise or Standard Edition (OBIEE) to access the data that made use of the OLAP features. The pure relational option just depends on whether we choose to deploy only the data objects and not the OLAP metadata. In reality, most people choose either the pure relational or the multidimensional. If they want both, they implement separate data marts. In fact, the default when creating dimensional objects and selecting relational for the implementation is to only deploy data objects. This case would allow us to use the dimensional

objects to load the data warehouse without needing to deploy OLAP catalog objects representing them. Tools like OBIEE or Discoverer can still derive Business Intelligence objects for dimensional oriented models in those tools using just these relational dimensional objects in the database.
Just to be clear, does all this mean that if we haven't paid for the OLAP feature for our database, we can only design our data warehouse using the relational objects; and therefore must our decision to design dimensionally change? The answer to that would be an emphatic no, since we just mentioned how OWB will let us design dimensional objects, cubes and dimensions, and then implement them physically in the database as relational objects. The benefit is that the same dimensional design can be implemented at a later time in an OLAP database just by changing a single setting. There are features of the Warehouse Builder for handling dimensional features automatically for us, such as levels, surrogate keys, and slowly changing dimensions (all of which we'll talk about later) that designing dimensionally provides us. We would have to implement these manually if we designed our own tables. Most people who use the Warehouse Builder will use it in that way, so we'll definitely want to make use of that feature to maximize the usefulness of the tools to us. This provides us with flexibility and it is the way we are going to proceed with our design. We'll design dimensionally using a cube and dimensions, and then can implement it either relationally or dimensionally when we're ready.

Summary

We have now gone through the process of designing the target structure for our data warehouse. We began with a very high-level overview of data warehouse design topics, then talked about dimensional design and the relational versus multidimensional implementation, and then we discussed the differences between them.Our design for ACME Toys and Gizmos is very rudimentary, just to give us an introduction to designing in OWB. You'll want to read in more detail about design when you tackle a real-world design because you may run into other issues we didn't have time or space to cover here.
We're going to actually implement the design in OWB in the next topic.