Database Design Using JDeveloper

By Susan Duncan, Oracle Corporation

In the last issue of the Journal, I gave an overview of much of the database functionality that lives in JDeveloper. This time I want to deep dive into how you can use one logical database design to produce physical database models for different databases with differing physical characteristics. In part 1 I explained how JDeveloper uses a UML class modeler to produce a Logical model. Taking that UML integration one step further I will show you how to customize your model such that it can be transformed to, say, a MySQL database as well as an IBM, TimesTen, and many other databases including, of course, multiple Oracle flavors too.

The Example

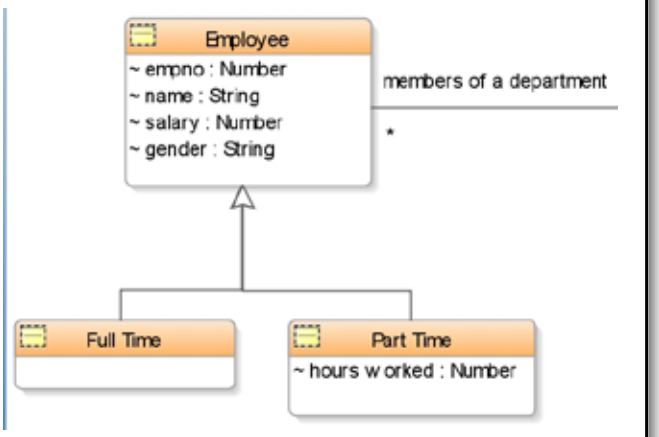

Illustration 1: Part of the logical model

In illustration 1, we return to the logical model created in part 1. This extract shows the Employee class and UML generalizations to specify the two sub-types of Employee: Full Time, Part Time, and an association to Department. In the previous part of this article, I explained how this could be transformed to a physical Oracle database table as shown in Illustration 2.

But what sort of improvements could be applied to this table? Many organizations have standards that need to be adhered to. The table has been correctly pluralized to EMPLOYEES, but what if the standard was something like EMP? A primary key column EMPLOYEE_ID has been created, but maybe the EMPNO is unique and could be used as that column. The Foreign Key name is long and doesn't conform to a standard. NUMBER columns have no precision or scale. All the VARCHAR2 lengths are 20, but they all need differing lengths. The schema has defaulted to the JDeveloper project name (PROJECT3). And what if the physical database was to be MYSQL: NUMBER and VARCHAR2 are not valid data types?

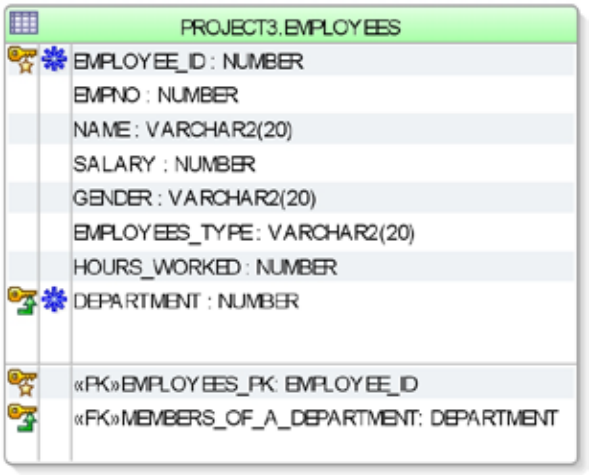

Illustration 2: Physical model of table EMPLOYEES

All of these things could be edited manually in the physical database model or models in JDeveloper, but there is a better approach – to set these things up at the logical level so that any standards are adhered to and can be used over differing physical models.

THE APPROACH

As the UML Class modeler is being used to model the logical database, this means that any changes that are made to the class model could possibly affect other non-DB modeler team members. Perhaps they are using the same model to generate Java or ADF Business Components. So any changes you make must not invalidate any of the other modelers' requirements.

To overcome this, you are able to use a UML concept called a UML Profile. This allows you to apply a profile to the class model that is used only when you transform the model to a physical database model. The ability to apply and use UML Profiles in JDeveloper is planned for the 11.1.2 release.

The Database Profile that I am going to example in the rest of this article is planned to ship with 11.1.2. The first step is to apply the profile to the UML model. A profile is applied at the UML package level. This might sound very non-DB oriented, but it is simple in JDeveloper. In the Application Navigator are listed all the UML elements you have created. Select the package.uml_pck file and open its property dialog. As shown in Illustration 3, select Profile Application, click the Add button, and select DatabaseProfile to apply it. Now you are ready to use the profile on your logical database model. A profile is made up of stereotypes that can be applied to different elements of your model.

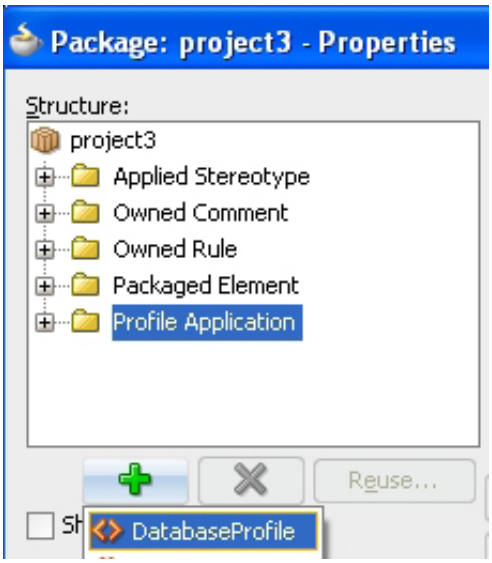

Illustration 3: Applying the DatabaseProfile

THE STEPS

Table Name

The first thing you want to do is ensure that whenever the Employee class is transformed to a physical DB table it is called EMP, not the default EMPLOYEES. To do this you apply the stereotype from the profile to the class name. Double-click the Employee class to open its properties. Select Applied Stereotype and click the Add button. What you want to do is apply a stereotype to the transformed database class name – so select Database Class from the drop-down list. In the Name after Transform value enter EMP. It's that simple: now any time the Employee class is transformed to a table, it will be called EMP.

Column Attributes

The same principle is used to apply a stereotype to a class attribute – that will transform to a column. Select the attribute in the diagram and double click to open its UML property palette. Select Applied Stereotype and click the Add button. This time there is no dropdown to select from, but a list of properties is available to you.

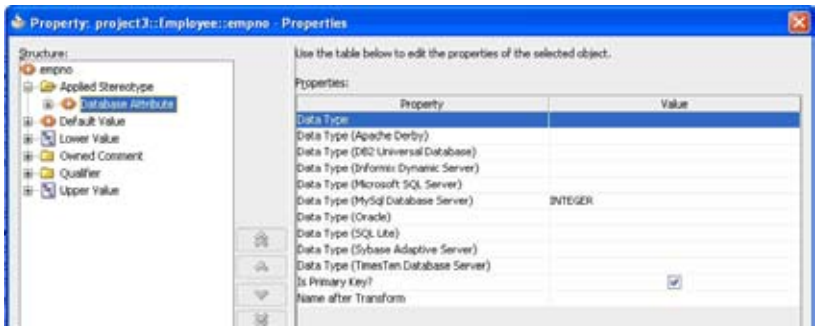

Illustration 4: Database Attribute Stereotype properties

In the empno attribute example in Illustration 4 (starting at the bottom of the list) – EMPNO is the column name you want after transform, so no need to enter a new value there. But this is also the column that should be used as the Primary Key – so check the box. Now, what about the datatype for this column? If you enter no data types here then all Number attributes will become default NUMBER columns. If you know that the same data type can be used over many different databases, then apply it to the Data Type property. But if you have a specific datatype requirement for a database – you can override that value. In this example, every database type (rightly or wrongly) will have an EMPNO column of type NUMBER except the MYSQL database (INTEGER). The same principle applies to precision and scale: NUMBER(6,2) or FLOAT(600,4000) can be applied as needed to numeric types.

And it's the same with Strings: VARCHAR2(100), VARCHAR(255), CHAR, and so on are all valid data types. One thing to remember – the transformer will use whatever you enter. It is not doing any semantic checking, and neither is the physical database model on transform. So an error of VARCHAR(400) would appear in your physical model. It will be caught when the SQL is validated.

So, you can go into every attribute in your class model and hand-edit it to apply the correct stereotype. But that doesn't sound very efficient and there is a way around that too.

Primitive Types

Take the gender and name attributes as examples. One way to populate GENDER that is widely used is as 'M' or 'F' – a one character column. You might have a standard in your organization that the default length of columns like NAME is 25. How can you most easily apply these to all the appropriate attributes in your model – without having to go into the UML properties and apply a stereotype to each one individually? (See Illustration 5)

Let's take the gender example first. In JDeveloper 11.1.2, it is planned, in addition to shipping the Database Profile, to ship a library of Primitive Types that can be used in a class model. Illustration 5 shows the properties of the gender attribute. Note that the Database Stereotype hasn't been applied, this are the standard UML properties of gender. Selecting the dropdown list of the Type value field opens the shipped Primitive Types library (Illustration 6). For gender, using DatabaseString1 is a good choice. And as you get to know the types in this library, you will not need to refer to the library as described here; you can use the in-place edit in the diagram to type in DatabaseString1. (See Illustration 6.)

But what about Primitive Types that are not shipped in the library? You can create your own using the Applied Stereotype route again. In your class diagram, create a new Primitive Type, for example String-25Type. Open its properties and apply the Database Datatype stereotype. Now complete the properties of your string for the databases you use (similar to Illustration

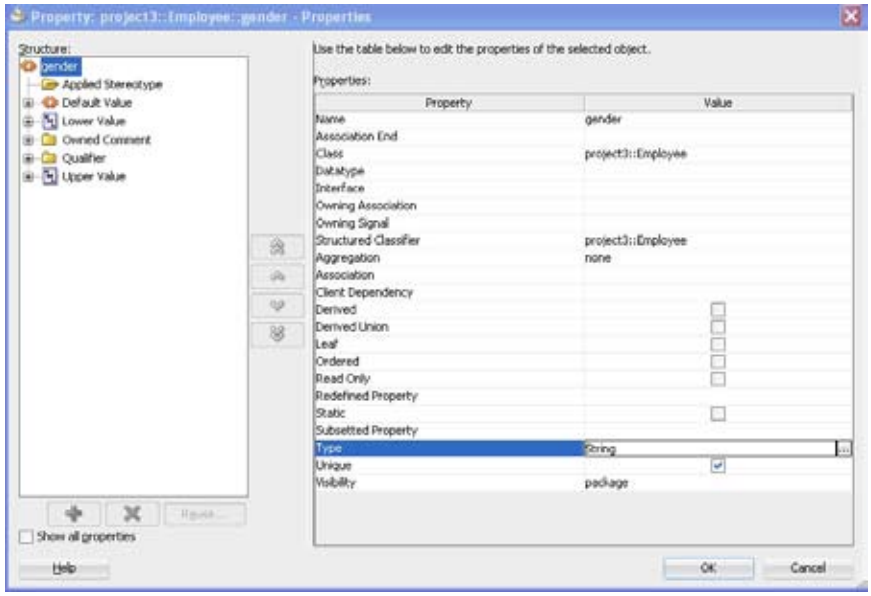

Illustration 5: Properties of gender

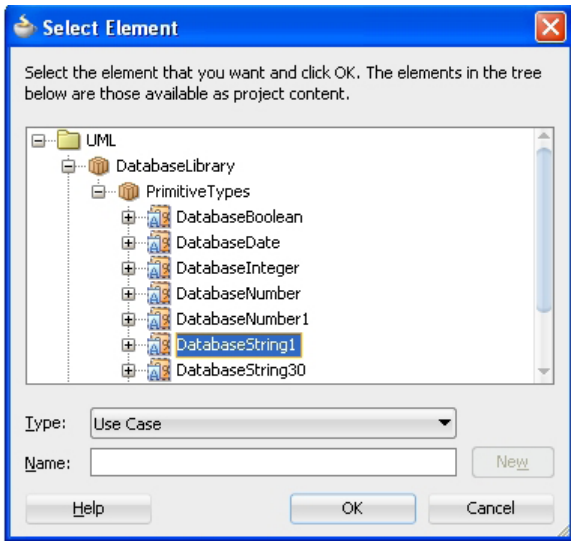

Illustration 6: Primitive Types library

5). Now you are ready to use your new type for the name attribute as shown in Illustration 7. On transform, the properties of the type will be used against the attribute creating the column you want.

Foreign Key Name

Another standard that is often enforced in an organization is how Foreign Keys are named. Applying the Database Association stereotype to the association between the two classes gives you a number of naming rule options such as basing it on both table names, UML name, or write-in name.

Schema Name

In the example here the schema name defaults to the package name in the project. Again, using a stereotype you can change that. Return to the package.uml_pck file in the navigator and open its properties. Apply the Database Package stereotype and you can name your schema for your transformed database.

The Transform Wizard

In the previous article, you learned how the transform wizard declaratively guides you through the transform options where, amongst other things, you can capitalize the name and insert underscores between words (part_time), attempt to pluralize names (EMPLOYEE to EMPLOYEES), and specify how subtypes are transformed (such as a discriminator column or separate tables). In 11.1.2, one important enhancement to the wizard allows you to specify the type as well as the name of any discriminator column that you specify for sub-types transformation.

THE RESULT

Illustration 8 shows the physical table definition transformed from the logical model in illustration 1. It takes into account the points raised in this article such as the Primary and Foreign Key names, the name, length, and types of the columns, and the Schema name.

This model is improved over the default transform by applying and using a UML Profile during transform of the model to a physical database model. The Database Profile is planned to be released in JDeveloper 11.1.2 along with a library of Primitive types.

The Database Profile enables the maintaining of certain database schema standards whether these are Oracle specific or across a number of different database types.

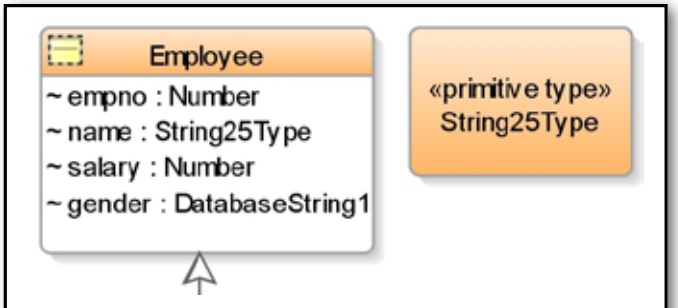

Illustration 7: Create and use a Primitive Type

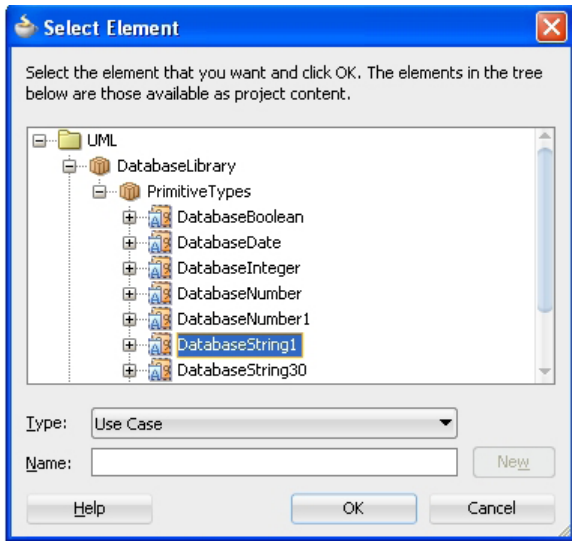

Illustration 8: Enhanced transformed EMP table

Oracle Development Tools Member

About The Author

Susan Duncan is a principal product manager in the Tools Development group at Oracle, specializing in Application Lifecycle Management, Team Development, and ADF. She joined Oracle UK Consulting in 1997 and worked on various projects worldwide designing, promoting standards, and implementing Oracle Designer and Forms Server applications. In 2001, she joined Product Management focusing on Oracle Designer and UML and Web services in JDeveloper. More recently she has focused her attention and skills on versioning systems, database modeling, and application lifecycle management, including the development of Oracle Team Productivity Center, and working with the open-source CI system Hudson.

Kscope's Gone Mobile

Kscope11? Yeah, there's an app for that!

Not a part of the iPhone crowd?
We've got you covered.

iPhone/iPad/iPod
Visit the App Store from your device. Search for ODTUG Kscope11.
Download the free app and start exploring!

Android, BlackBerry, other smartphones
Visit kscope11.gatherdigital.com from your mobile device. Do not put in any "www's." If you try to get to this link from a PC, it wil not work. It can only be used from a mobile device.

Everything you need to Know in the palm of your hand

Know where to go:
Use the events feature to browse everything going on at Kscope11. Then select which events, sessions, etc. you would like to attend and create a "My Schedule." Listings include a map so you won't ever get lost, abstract and presenter information, and much more. The hardest part about attending Kscope is trying to determine which sessions you should attend, and now you can hold that information in the palm of your hand.

Know who to see:
If you are looking to connect with an exhibitor on site, we make that easy. In the weeks leading up to the event, peruse the listings, check out the exhibitor offerings and special events, and add them to your schedule.

Stay connected:
At Kscope11, ODTUG will push out messages providing you with updates on sessions, events, and much more.

Provide feedback: While on site, you will be able to use your mobile app to provide feedback on the sessions you attend. Kscope is your conference and your feedback is greatly appreciated and now super easy to provide!

Kscope's Mobile App is just one more way ODTUG is leading the way to the most interactive, exciting event yet!