ORACLE White Paper

An Introduction To Designer/2000 For DBAs

VERSION 1.0: OCTOBER, 1995

Part C10483

Part C10483

October 1995

Author: Kimberly Free, Oracle Corporation

Contributors:

Table of Contents

Introduction

This paper will discuss how DBA's can use the Entity Relationship Diagrammer, Data Diagrammer, Database Design Wizard, Table to Entity Retrofit Utility, Design Recovery Utility and Reconcile Utility to create, modify and manage database objects (tables, indexes, roles, tablespaces, etc.). The Repository Object Navigator and Reporting capabilities will also be discussed from a DBA's perspective.

Specific guidelines for using the Designer/2000 toolset in designing and implementing a new database will be presented as will how to introduce new objects into an existing database. Techniques that can be used to reconcile the on-line Data Dictionary to the Repository, and vice versa, will also be discussed.

Before you start

Before you can begin entering data in a Designer/2000 Repository, there are some planning and administration issues that need to be addressed. The first of these is how to organize the data in the Repository - will you have a single application containing all data items for the enterprise, one application in the Repository for each application in the enterprise, or some variation of these extremes? A discussion of how to determine the way to organize your Repository is beyond the scope of this paper. Suffice it to say that some thought should be given to organizing the Repository before you begin entering data into it.

Another area that needs to be addressed is the role of the Designer/2000 Repository Administrator, also known as a CASE administrator. Some organizations have an individual who is dedicated to performing this task, while others share this responsibility among several people. Addressing this issue is important for two reasons: 1) the processes to be used in administering the Repository (e.g., when is an application frozen) will vary depending upon how this role is implemented and 2) the access rights granted to an administrator of the Repository are very powerful, which means they should be given to a limited number of users.

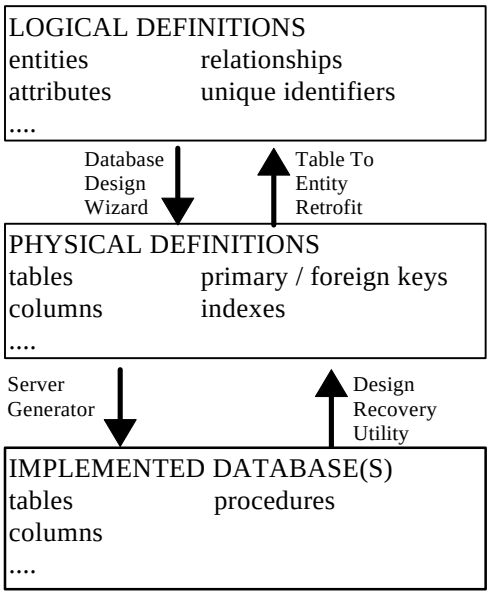
The issues surrounding the processes to be used in managing a Repository need a paper of their own to be covered appropriately. Therefore, this issue will be bypassed and granting access to the Repository will be discussed instead. Access to the Repository can be given to explicit users, it cannot be granted to roles. When the CASE administrator uses the Repository Administrator Utility to grant access to the Repository, they must select an existing Oracle user and then determine if the user is a MANAGER or USER, as well as if they should have READ or WRITE privileges. The meaning of these privileges are self evident - MANAGERs are Repository administrators, USERs can access the Repository data and READ or WRITE (which implies READ as well) the data it contains. It is recommended that users be given the minimum privileges needed to perform their job. For example, if you want a functional user to do on-line reviews of diagrams you should grant them USER, READ privileges while a DBA who is not a Repository Administrator should be given USER, WRITE privileges.

Once these issues have been addressed, an authorized Repository administrator needs to use the Repository Object Navigator to create the application(s) which will contain the information on entities, tables, users, etc. No actions, aside from creating Repository users, can be performed until an application is created in the Repository. The administrator also needs to grant access to specific applications for each authorized Repository user. At this point, if the first steps to be performed are to reverse engineer an existing Oracle database, a Repository user should create the databases in the Repository that the recovered objects will be assigned to. As the repository fully supports distributed databases, all databases which contain tables or snapshots used in this application which will be reverse engineered should be created.

Introduction to the Repository

Before discussing some techniques for utilizing the Repository in the day to day tasks performed by a DBA, a brief introduction to the structure of the Designer/2000 Repository is needed. The easiest way to explain how the Repository is structured is shown in Figure 1, which illustrates how the data contained in the Repository can be placed into one of three layers: logical definitions, physical definitions, and implemented databases.

Figure 1 - Conceptual structure of Designer/2000



The logical definition layer contains information that is typically considered the logical model of a system (e.g., entities, relationships, functions). The physical definition layer contains information on what is typically considered the physical model of a system (e.g., tables, screens, menu structures, user access). The implemented databases layer in Figure 1 represents two important pieces of information. First, it highlights that information in existing databases can be loaded into the Repository. Additionally, the repository tracks when items have been generated from the Repository so if you try to regenerate them you are warned they have been generated and are asked if you would prefer to regenerate instead. This second feature is more geared towards applications developers generating forms, reports, etc., but it can also be useful to DBAs.

The implementation of the Repository structure enables the Repository to support many different software lifecycle development methodologies. For example, the logical portion of the Repository is the first area to be populated if a traditional information engineering approach to software development is followed, e.g., entity relationship diagrams are created to model the data requirements. The physical level is the first to be populated in a rapid application development

environment, e.g., the tables and keys needed to contain and access the data required to support the application. In the event you wish to load information from your existing Oracle database into the repository, also known as reverse engineering, the Repository Reverse Engineer DDL utility can be used. The strength of the Repository is that data can flow ‘up’ and ‘down’ the structure as needed, allowing the reuse of data input at one stage of the software development lifecycle as appropriate in the next or previous stage.

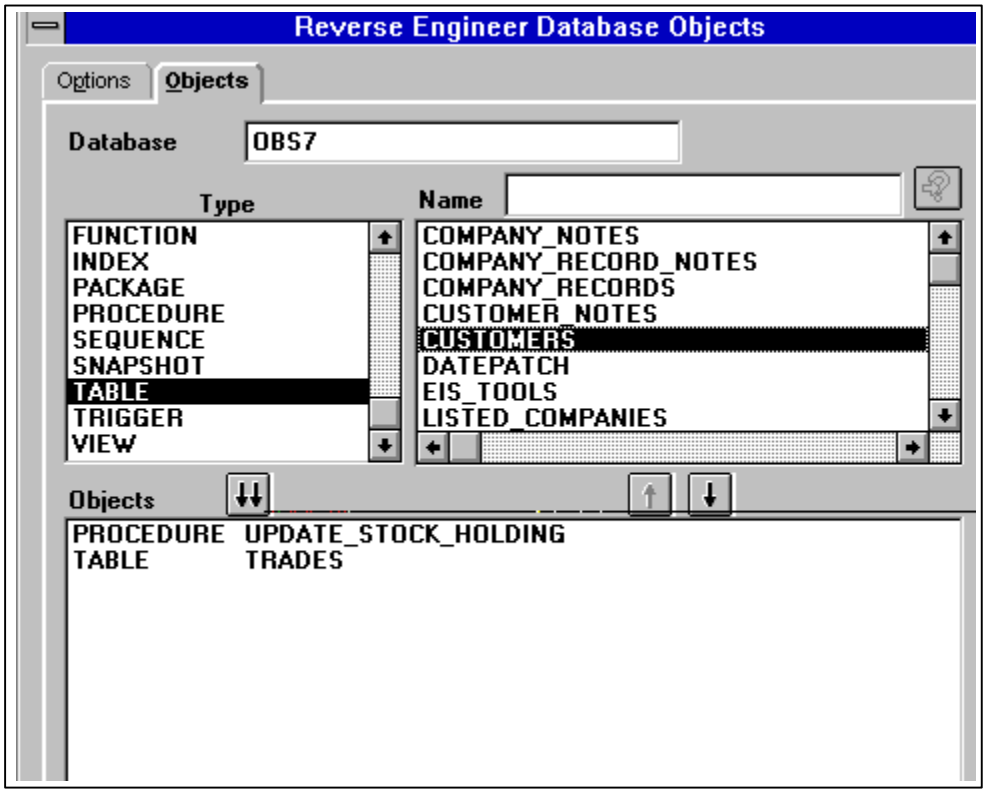
Initially Populating The Repository

Now that we have a fundamental understanding of the Designer/2000 Repository structure and how it supports multiple development methodologies, it is time to begin inputting data into the Repository. How this is accomplished varies depending upon whether you start with an existing Oracle database, a new system or a logical model contained in another CASE Repository.

From an existing Oracle database

For many DBA's, their first use of the Repository will be to support an existing Oracle database. To populate the Repository, the DBA will use either the Repository Object Navigator or the Data Diagrammer to invoke the Reverse Engineer DDL utility. The database defined in the Repository which will contain the recovered database objects needs to be identified, as well as the user id, login and connect string to the database containing the objects you are interested in. Once this information has been provided, you indicate as many or as few database objects as you would like to recover. The utility will access the appropriate information in the on-line data dictionary and populate the physical definition layer of the Repository with information on those database objects you selected. See Figure 2 for the Design Recovery interface.

Figure 2 - Reverse Engineer DDL Utility

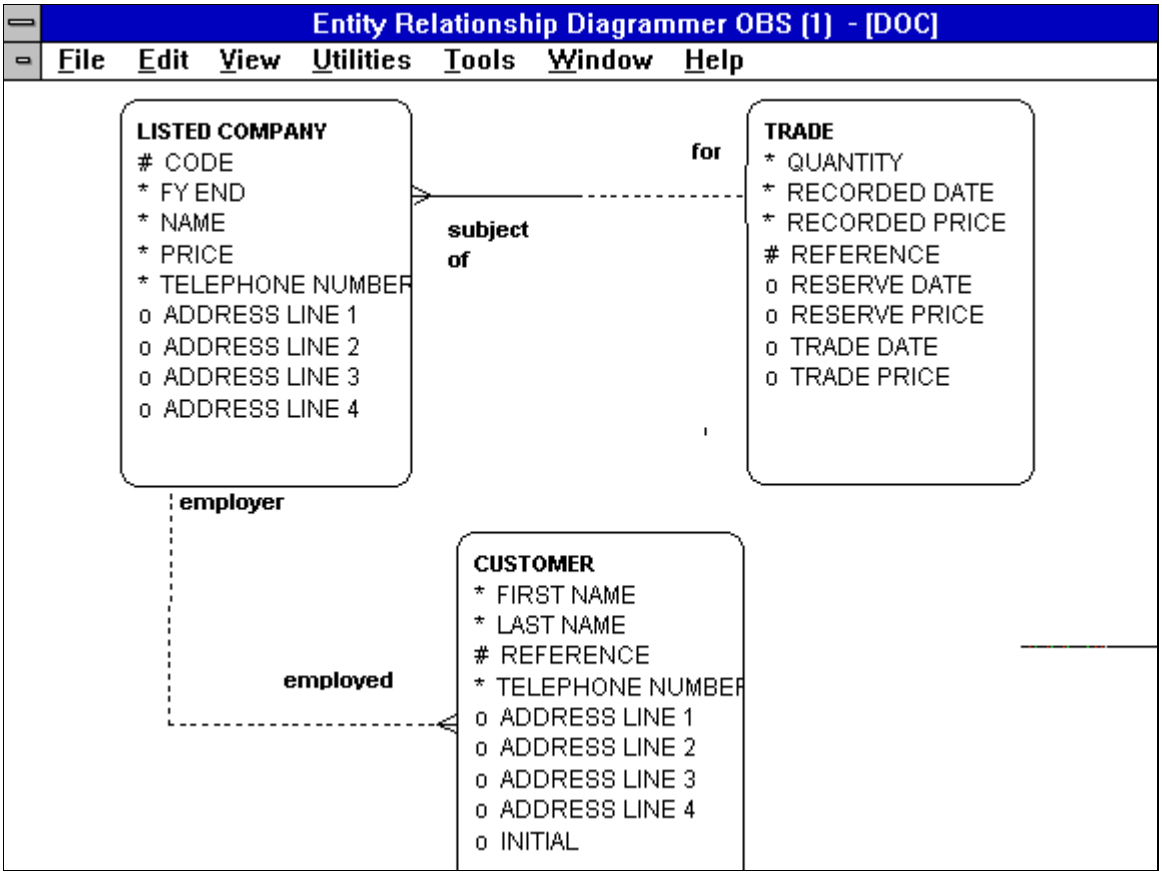


At this point you can use the Data Diagrammer to create diagrams showing the recovered database objects or the Table to Entity Retrofit utility to push the newly recovered physical items ‘up’ to the logical definition layer. Once the Table to Entity Retrofit utility is invoked, a comparison is done to see which tables in the Repository are mapped to entities. All those tables which are not mapped are candidates to be mapped and the DBA can select which ones they want to have the mapping performed on. For those tables selected, a one to one mapping will occur: tables become entities, columns become attributes, relationships are built based on primary and foreign keys. At this point, the Repository has been populated with the requested objects from an existing Oracle database. This information can now be used to maintain, enhance and/or document the database.

With a new design

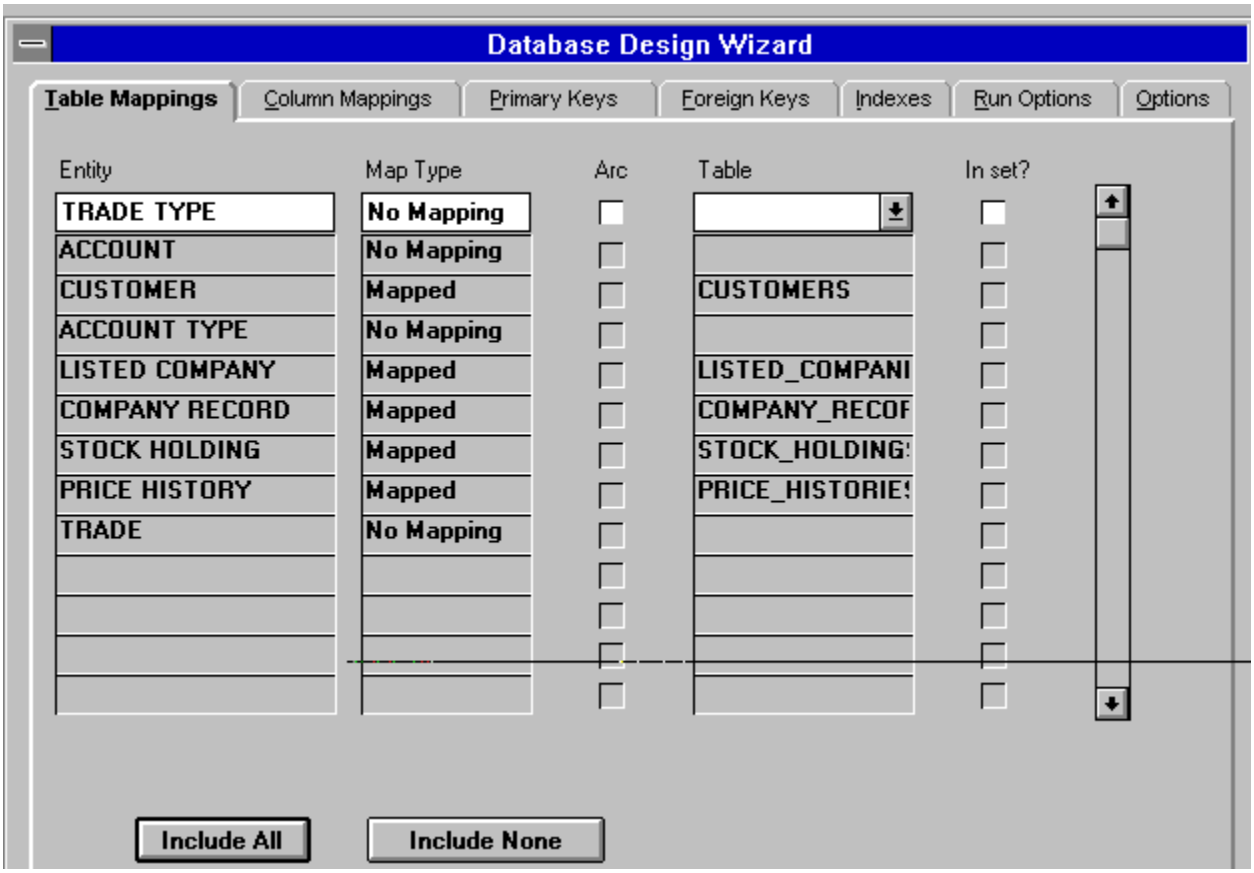
DBA’s who are creating a new database have two approaches that they can use, depending upon the software development approach used in their organization. For those who follow traditional information engineering methodologies, they (or their data administrators) will model the data requirements using the Entity Relationship Diagrammer. Domains, which carry both logical and physical characteristics, can be created in either the Repository Object Navigator or the Entity Relationship Diagrammer. Refinements to entities, attributes and/or relationships (e.g., changing the length of a character field) can be made in the Entity Relationship Diagrammer or using the Repository Object Navigator. The Repository Object Navigator is especially useful if you want to make the same change to multiple objects (e.g., assigning a domain to several attributes). A sample entity relationship diagram is shown in Figure 3.

Figure 3 - Entity Relationship Diagram



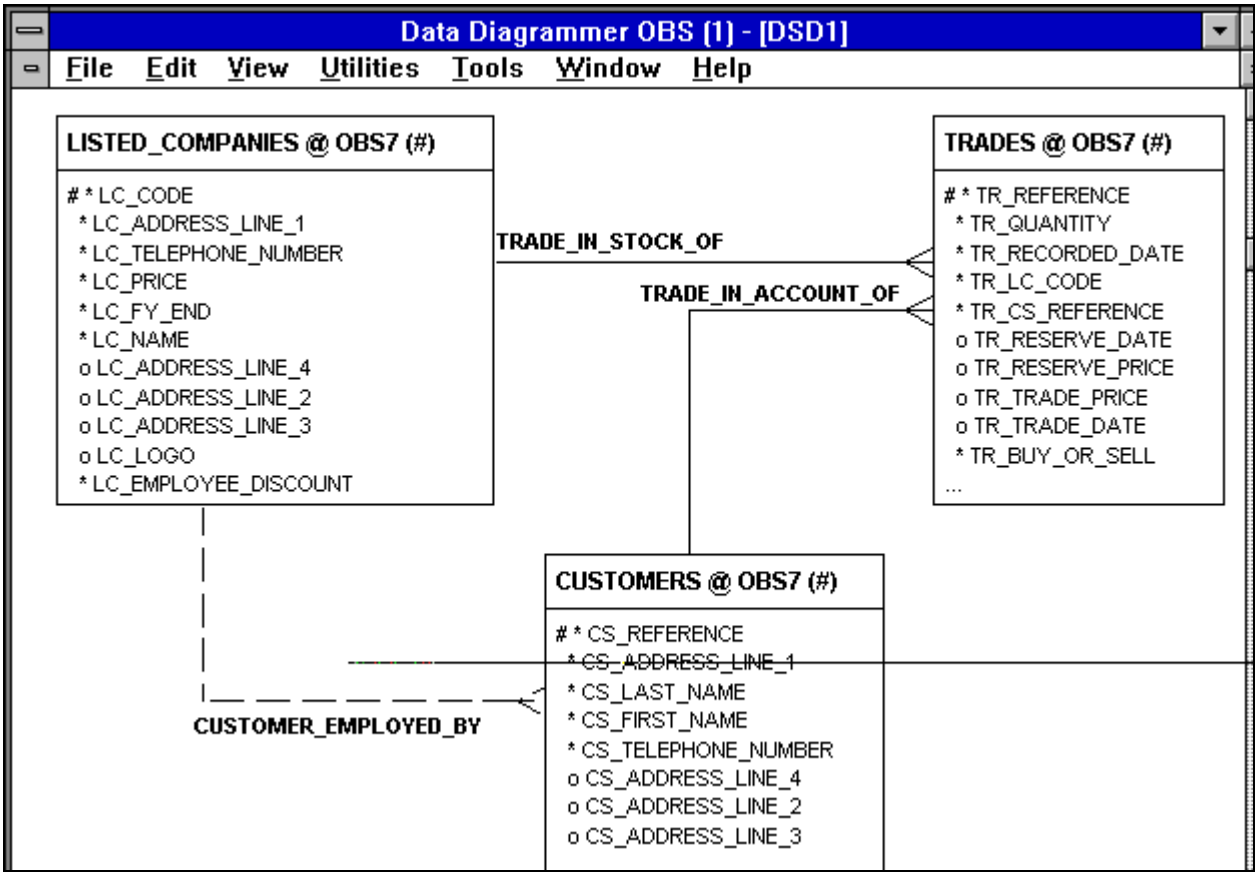
Once the logical model is somewhat finalized, the Database Design Wizard can be used to create a draft database design in the physical definition layer of the Repository. This process allows the DBA to select which entities they want to migrate to tables. For those models which utilize super and sub types or arcs, options on how to physically implement these modeling techniques are available. Once entities are mapped, the DBA can then choose to map none, some or all of the related characteristics: attributes to columns; primary keys, foreign keys and unique identifiers to indexes. The DBA also has control over when to commit these mappings to the Repository, allowing full control over a very iterative utility to do a draft translation of a logical model to physical design. This utility can also be used as maintenance and enhancement activities take place on a data model over an applications lifetime. The Database Design Wizard is shown in Figure 4.

Figure 4 - Database Design Wizard



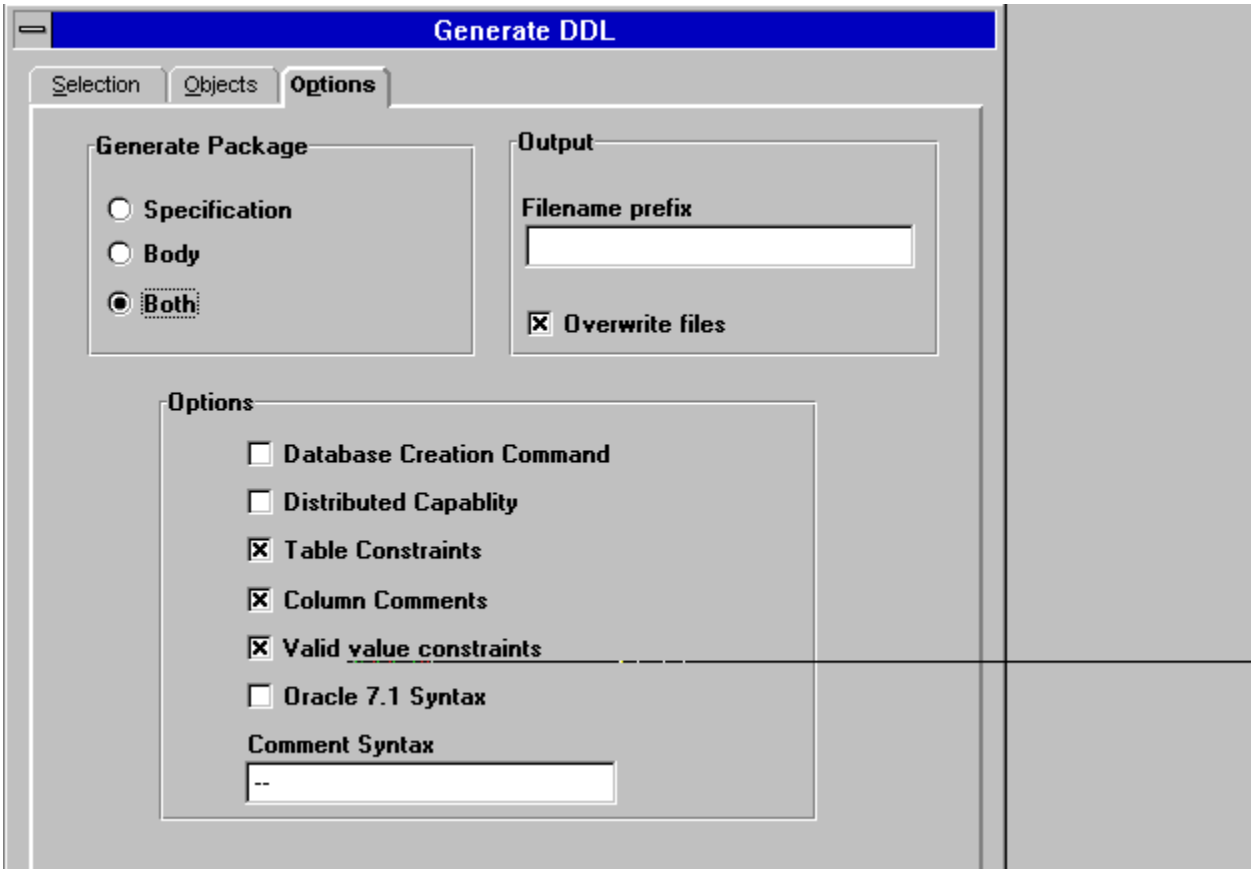
For DBA's who are involved in rapid application development projects, the Data Diagrammer will be used to create the tables, columns, views and snapshots needed by the application. Refinements to tables, views and columns can be made in the Data Diagrammer or using the Repository Object Navigator. The Repository Object Navigator can also be used to record information on tablespaces and rollback segments as well as the datafiles used by each. Data on database roles, users and which users are assigned to the specified roles can also be stored in the Repository. See Figure 5 for sample Data Diagram.

Figure 5 - Data Diagram example



Regardless of which technique is used to enter information into the Repository, at this point the Generate DDL Utility can be used to generate the data definition language needed to create the database objects. Once again this is an iterative process, enabling the DBA to select one to all of the tables, indexes, users, roles, procedures, and other database objects the Repository stores information on. The DBA can even choose to have the Server Generator generate the 'create database ...' script or use Oracle 7.1 syntax. Figure 6 shows the options available in the Generate DDL interface, the interface to select the items to be generated is the same as that for the Reverse Engineering DDL utility shown in Figure 2.

Figure 6 - Database Design Wizard interface



From other repositories

Oracle offers CASE*Exchange which allows the bi-directional exchange of logical data (entities, attributes, relationships, unique identifiers and domains) between Oracle and selected other repository vendors. The use of this tool will populate the logical definition level of the Repository, where the DBA and/or data administrator may refine the logical design. Once the import from CASE*Exchange has been completed, the DBA proceeds in the same manner as if they or the data administrator had entered the data themselves using the Entity Relationship Diagrammer. For more details of the Oracle Exchange product please refer to the white paper 'Oracle Exchange 4.0 - Technical Overview' (Part no. C10407)

Oracle also offers a migration path through the Repository Administrator Utility for users of its CASE v5.0 and CASE v5.1 tools. This migration path moves all information for a specific application

(logical and physical) from the older Oracle CASE repository into the Designer/2000 repository. Again, once the data has been loaded into the Repository the DBA can proceed as if it had been manually input using the Entity Relationship Diagrammer, Data Diagrammer or Repository Object Navigator.

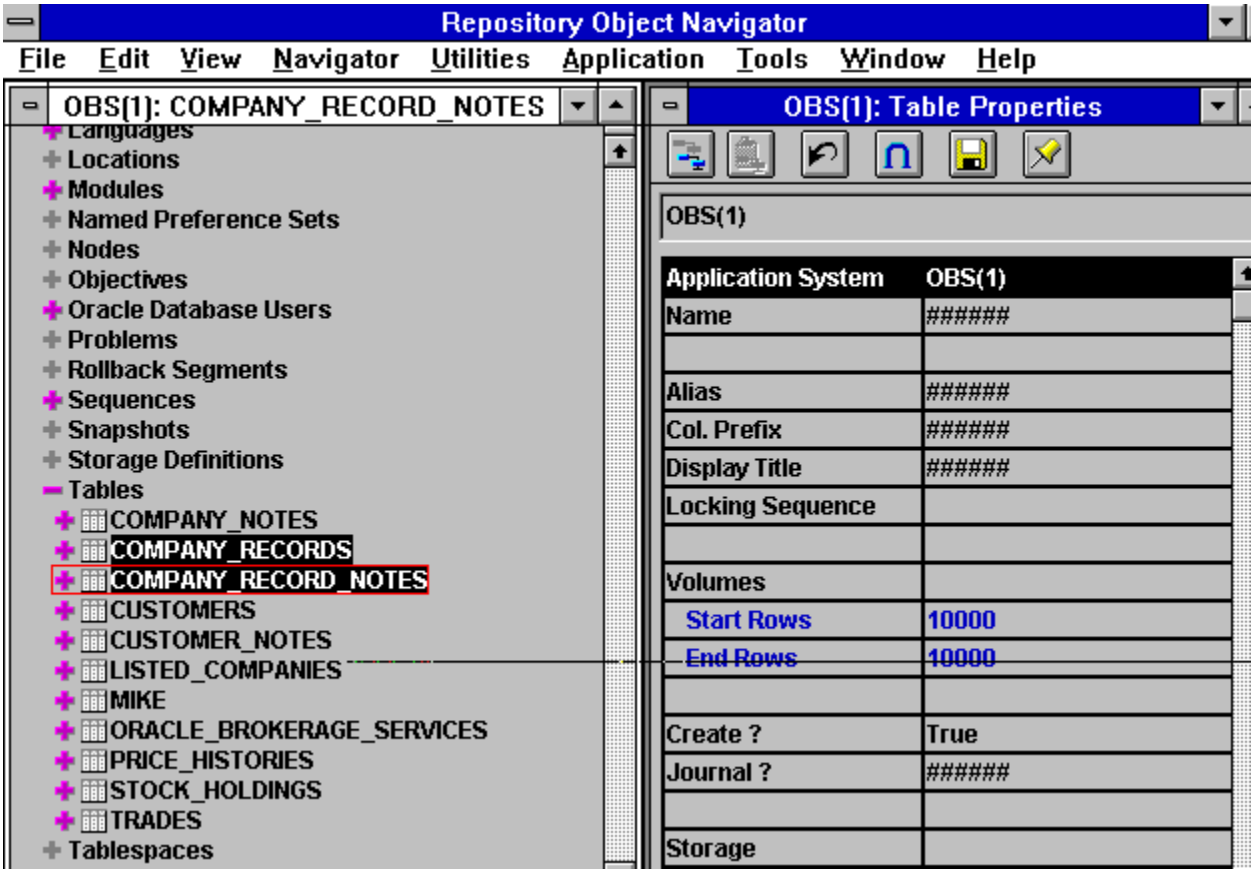
The last way to input data from a non-Oracle repository into the Designer/2000 Repository is through the use of the Designer/2000 API. This is a series of PL/SQL packages that can be used from any 3GL or 4GL to populate the Repository. The effort required to populate a Designer/2000 Repository using this technique can be substantial, but for an organization with a lot invested in an enterprise wide model in a repository not supported by CASE*Exchange, the effort to write the programs to populate the Repository is worth the ability to access the corporate model. As before, once the Repository has been populated it can be updated using the Entity Relationship Diagrammer, Data Diagrammer or Repository Object Navigator as appropriate. For further information on the repository API please refer to the white paper ‘Designer/2000 Repository API - Technical Overview’ (Part no. C10408)

Maintaining the Repository

Once any repository has been initially populated with information on a data model or database, there are numerous issues relating to the maintenance of the data in that repository. The Designer/2000 toolset provides several features to assist in these maintenance activities.

Adding information to a data model or database design in the Repository is a straight forward activity as long as the model or design being worked on is the initial implementation. Once a data model has been passed through the Database Design Wizard to create a default database design or the database has actually been implemented, the issues surrounding maintaining the Repository increase. Additions, deletions and/or changes to the Repository can be made using the diagrammers (Entity Relationship, Data Diagrammer) or the Repository Object Navigator depending upon the information to be modified and the DBAs personal preferences. Many DBAs prefer to use the diagrammers when creating new tables and foreign key relationships while using the Repository Object Navigator to modify existing information in the Repository, especially if that information is regarding sizing. Figure 7 shows how the Repository Object Navigator can be used to change the initial size of several tables defined in the Repository.

Figure 7 - Using the Repository Object Navigator to modify the sizes of tables in the Repository



Default database design exists

Changes made to the data model after the Database Design Wizard has been run can be easily incorporated into the physical definition stored in the Repository. This is done by running the Database Design Wizard again. There is a tab on this utility that allows you to decide what can be changed once the initial default database design has been done (e.g., columns, keys, indexes, volumetrics). The DBA selects what Repository characteristics can be changed. Once this has been done, the modified logical items are identified, at the table or specific object type tab by clicking in the appropriate check box. The utility is then run and the physical definition level of the Repository is updated with the changes made in the logical definition level. The database object tabs as well as the run time options tab allows the DBA tight control over what they would like to transform from the logical model to physical design.

You can generate DDL using the Server Generator as many times as you like. The utility prompts you for the objects to be generated and then it accesses the Repository to determine the characteristics of those objects. The important thing to remember is that you must run the Generate DDL Utility after making changes to the physical definition in the Repository. It is also important to note that the DDL generated always assumes that the object being created does not already exist in the target database, so if your day to day DBA procedures include a DROP, CREATE technique to introduce changed tables, indexes, etc. you will need to manually perform the drop or modify the generated scripts to do the drop.

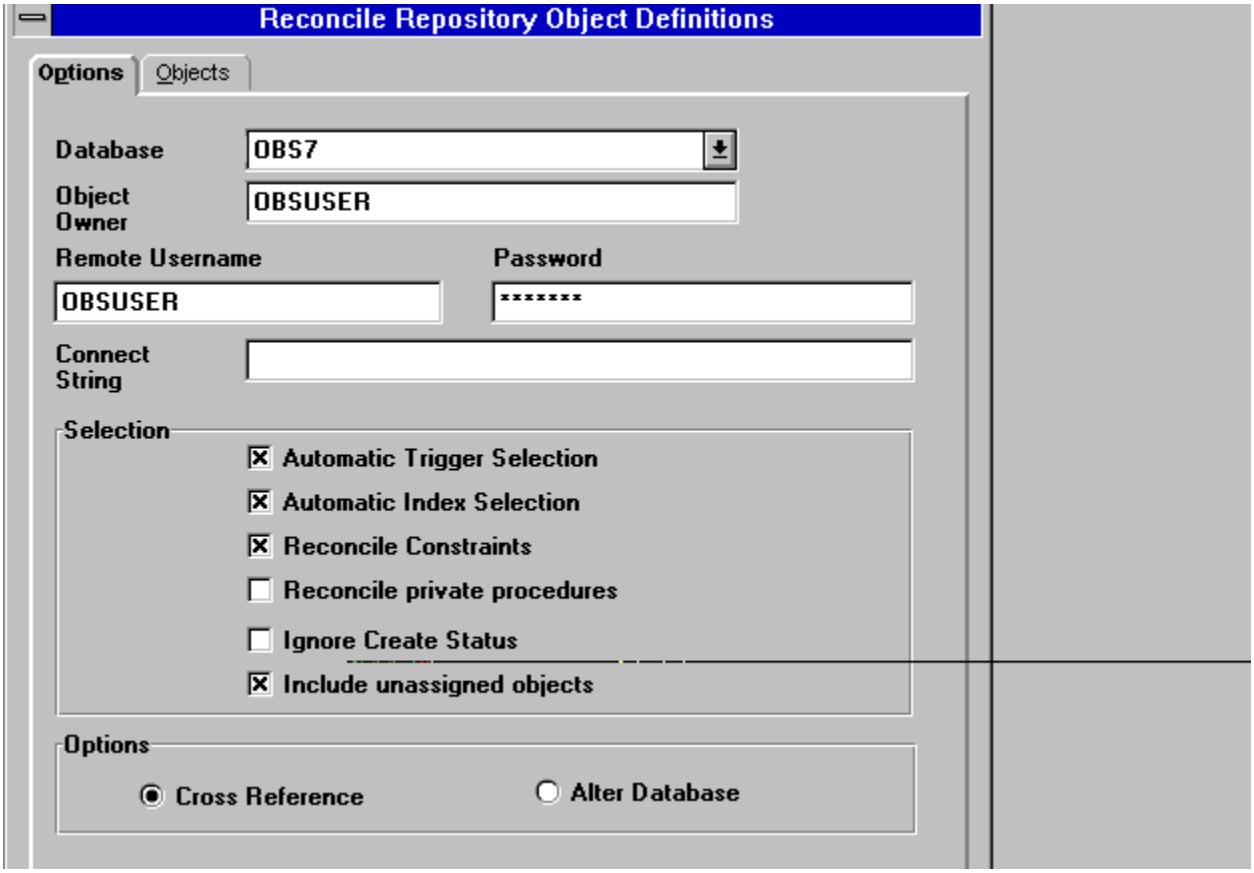
Database implemented

The trickiest area of Repository maintenance occurs once a database has been created using the data you have input into the Repository. At this time, DBA's must worry about keeping the Repository and the on-line Data Dictionary synchronized from two perspectives: 1) what items have been added or dropped from the Repository and/or on-line Data Dictionary and 2) what items have been altered in the Repository and/or on-line Data Dictionary.

One of the most important utilities to assist in this process is the Retrofit Utility. This utility allows a DBA to compare the definitions of a database in the Repository to the actual on-line Data Dictionary of the Oracle database. Once this report has been run and differences identified, the DBA can determine whether the Repository needs to be updated, the actual database modified to reflect the contents of the Repository or a combination of the two performed. The first of these is achieved using the Reverse Engineering DDL utility, as discussed in an earlier section of this paper.

The second option is accomplished via the Reconcile Utility's ALTER DATABASE radio button option. The Reconcile Utility will create the SQL DDL needed to modify the database so that it will resemble the Repository before providing the DBA the opportunity to review the generated SQL and/or run it against the database. The last option is understandably the most difficult of the three. Based on the output of the Reconcile Utility report, the DBA will need to manually update either the Repository or the actual database (or both) and then use the appropriate technique already discussed to synchronize the remaining information. Figure 8 shows the Reconcile utility.

Figure 8 - Reconcile Utility



Additional maintenance issues

There is an entire area relating to maintaining the Repository that is beyond the scope of this paper - using the versioning and sharing capabilities of the tool to optimally organize and manage the Repository. Numerous philosophies exist for setting up multiple applications within a single Repository and determining whether or not and how to share Repository objects among them. Likewise, procedures should be developed on when a specific version of an application should be frozen in the Repository and a new version created. Many large organizations create a new version of the Repository when they are introducing major database changes into their application, freezing the current version for historical purposes (past versions of the application can still be accessed if they are unfrozen). Some smaller organizations never bother with version control on their applications, so each organization needs to determine what technique works best for them.

Using the Repository to create documentation

There is an often quoted saying which states that 'a picture is worth a thousand words'. This is especially true in the realm of database documentation, where DBA's rarely find the time to keep this important information up to date. Using the Designer/2000 Repository to create logical and/or physical models of your data provides you with a mechanism to easily create diagrams showing your entire enterprisewide model or various subsets of that model. These diagrams can then be used as standalone documents or embedded in other OLE2 compliant word processing programs to provide the required level of database documentation.

Additional documentation can be provided using any of the hundreds of reports provided with the Designer/2000 toolset. These Repository Reports are created in Oracle Reports 2.5. This enables Repository users to easily create new and customize existing reports (it is suggested that the original Oracle supplied source be copied and modified rather than modifying the original source provided with the product). Unfortunately, at this time only the reports which come with the Designer/2000 product can be called directly from the toolset, so customized and original reports need to be run outside the tool.

The following reports are especially valuable as a documentation mechanism (the titles are as they appear in the Repository Reports utility):

- *Entities and their attributes* - lists entities, their attributes and related characteristics
- *Relationships* - lists all entities and their relationships in sentences, useful for verifying/reporting on the data model
- *Definition reports for tables, views, indexes, tablespaces, rollback segments, etc* - reports the characteristics of database objects, useful for reviewing the physical implementation of the data model
- *Database objects* - listing of objects in the database by object type
- *Views and their table derivations* - lists views and what tables they are dependent upon
- *Database triggers* - information on triggers in the database, including whether or not they have been created and enabled
- *Tables and their Indexes* - listing of tables and the details of indexes on them

If you want to see if a specific report provides the information you are interested in before you run it, use the on-line Repository Reports Help utility and make the following selections:

1. Repository Administrator
2. Repository Reports
3. search for 'reports'
4. show topics
5. select 'Report Descriptions'

This provides a user friendly way to determine the information on each report provided with the Repository.

Additionally, since the Repository data is stored in an Oracle7 database, new reports can be written using any Oracle SQL compliant data access product.

One final note on using the Repository to document your database and data model: the output is only as valid as the information input into the Repository. If you do not input descriptions, notes or comments on entities, attributes, tables, tablespaces, etc. then the documentation produced is no more than the characteristics of those database objects. The Reconcile Utility should be run on a regular basis for all databases that are subject to maintenance activities to ensure that the Repository and on-line Data Dictionary for the database are in sync. This will ensure that the information in the Repository for production systems is accurate, thereby ensuring that future work done on the database using the Repository is built on an accurate foundation, as is the documentation produced.

Other Useful Repository Features

The Designer/2000 toolset provides the ability to store the source code of stored functions, procedures and packages (which are commonly referred to as modules) in the Repository. For organizations which charge their DBA's with being responsible for moving these database objects into the production environment, this feature enables DBA's to track this source code in the same manner as they manage all other database objects. Modules are assigned to applications in the Repository, thereby being managed in sync with the related tables, indexes, etc. which are needed to support the modules. Inputting information on the source code for stored procedures, functions and packages can be done using the Repository Object Navigator or the Module Logic Navigator. The Repository Object Navigator interface offers a standard MS Windows based editor to input the data into the Repository whereas the Module Logic Navigator provides a drag and drop interface. It also provides syntax checking before compilation occurs as well as some nice editing features specifically designed for PL/SQL scripts.

The table and index size report provides assistance is sizing objects in the database. For this report to have any value, volumetric information on tables (e.g., number of rows, growth rate) must be entered. Also, tables must be assigned to tablespaces in the Repository before running the report. The report is also intelligent enough to know that storage parameters assigned directly to a table take precedence over those assigned to the tablespace the table is assigned to.

There are numerous reports available to ensure the quality of the data in the repository. These are an easy and useful to check for oversights (an entity was created and the supporting detail omitted). For example, at the data model level, reports exist to indicate which entities have no attributes, no relationships, and/or no unique identifiers. Similar reports exist for the physical implementation of the database as well.

Last minute hint

Remember to put a backup and recovery plan into place for the database containing your Repository! The use of the Repository in software development projects is just as important to your company's success as the most mission critical application you will create using the tool - so protect the data in it accordingly.

Summary

This paper has provided some guidelines and techniques that can be used by DBA's in working with the Designer/2000 toolset. Various software development lifecycle approaches DBA's need to support were discussed, as was how to do ongoing maintenance of the Repository. Useful ways to use the Repository Reports were also presented.

The purpose of this paper was to introduce DBA's to some of the features and functionality available in the Designer/2000 toolset that can be useful to them in their day to day activities. There are many more useful features and techniques that can be used by DBA's in working with the Repository, but hopefully the material presented here provides a solid starting point for DBA's new to Designer/2000.