ORACLE White Paper

Designer/2000
for Oracle 7 Developers

VERSION 1.0: NOVEMBER, 1995

Part C10591

Part C10591

November 1995

Author: Andy Cutler

Contributors:

Introduction

The building of Oracle 7 based systems involves a number of major considerations including,

- Performance
- Integration of other application systems
- Documenting the system
- Communicating the design and usage
- Providing a scaleable architecture
- Avoidance of reinventing the wheel

The aim of this White paper is to examine how using components of the Designer/2000 product can significantly benefit in the tackling of the above challenges. The paper will cover in detail the following areas:

- Designer/2000 Product Overview
- Database Design
- Procedural server-side logic design
- Server Generation
- Maintenance
- Design Recovery

Designer/2000 Product Overview

Designer/2000 is Oracle Corporation's new generation modeling and generation toolset, and is rapidly becoming adopted as a valuable aid for development teams that are building and maintaining applications based on Oracle 7 technology.

One of the key components of the Designer/2000 product is the underlying repository. The repository is a controllable multi-user environment that provides for the design and generation of all Oracle 7 functionality. In fact the repository is built as a standard Oracle 7 database, and utilizes an API written in PL/SQL.

The Designer/2000 product covers the full spectrum of systems development from business process engineering through to delivery and maintenance of client/server systems. However, the brief of this paper is to focus on the design time environment for Oracle 7 developers, which includes the following products.

Data Diagrammer

The Data Diagrammer provides a graphical environment for developing and managing databases. Developers represent tables, views, keys etc. The Data Diagrammer creates, manages and displays logical or physical database designs.

The Data Diagrammer also offers the database designer:

- Database Design Wizard that creates normalized database designs directly from entity relationship models
- Facilities to specify client/server partitioning, allowing designers to cause constraints to be validated on the client-side, the server-side or both
- The ability to define data security, users, groups of users, roles and access rights
- Reverse engineering of database definitions and procedural components that were not created in the Designer/2000 environment
- Access to the Server Generator that transforms repository-based schema definitions into SQL scripts either for Oracle 7 databases or ANSI standard SQL databases

The Data Diagrammer delivers complete database designs, either from reverse engineered instance databases, from business models (entity/relationship) or directly entered designs. The Data Diagrammer can manage multiple diagrams, for example, to display partitioned database designs, but basing them on a single Repository. It can use colour, font and linestyle to customize diagrams, and the user can specify preferences for the appearance and content of diagrams that are retained for subsequent sessions.

Figure 1 illustrates the potential uses of the Data Diagrammer.

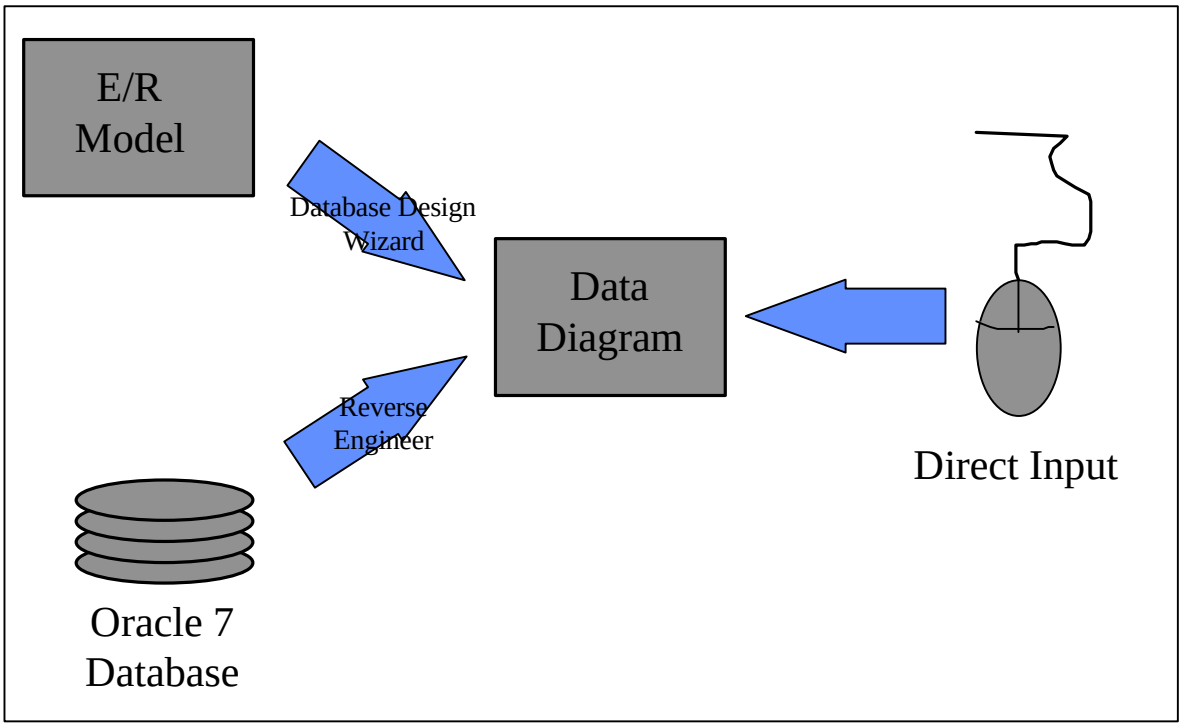


Figure 1. Data Diagrammer

Module Logic Navigator

Using Oracle 7, developers have the option to specify procedural logic to implement business rules on the server. These can be invoked as a result of pre-defined triggers on data changes, for example after insertion into the Customer table, or on specific application events, for example whenever the Customer Account Status is changed. The Module Logic Navigator is specifically designed to assist developers in creating and modifying such procedures.

A developer using the Module Logic Navigator can:

- Create any valid PL/SQL statement by drag and drop from a statement template library
- Reference any valid data definitions in the Repository by drag and drop
- Navigate large modules by using an auto-synchronizing outliner
- Find incomplete editing from the current or previous sessions
- Syntax check PL/SQL
- Create cross-reference documentation of which tables and columns a module accesses

PL/SQL modules created in this way can then be implemented using the Server Generator. And, of course, PL/SQL can be reverse engineered. The power of the Module Logic Navigator and the repository management tools are then available to help database administrators and designers get existing databases under control.

Repository Object Navigator

The Repository Object Navigator presents the developer with a Windows 95 explorer style interface to any and every object in the Designer/2000 Repository, arranged by type.

The Repository Object Navigator is not simply a read-only browsing tool. It provides update of all the objects, and allows the update of sets of objects disjointedly selected. Update can be either by direct input or from the paste buffer. For example, if a database column definition has been used in ten or twenty screens and reports, and the database designer changes that definition, it is essential to assess

the impact and make changes. Using the Repository Object Navigator the project manager navigates to the column changed, copies the changed property to the paste buffer, she or he then navigates to the set of modules that use that column, selects the ones which are to inherit the change and then pastes the new property value or values onto the target set. The change is effected simply and quickly, but still under user control.

Server Generator

The Server Generator reads definitions held in the Repository, including the specified distribution of constraint enforcement across the client/server divide and creates:

- Database definitions: tables, columns, views
- Keys and constraints for data integrity
- Indexes and clusters for performance
- Oracle 7 role-based security model
- Distribution design: nodes and databases
- Replication snapshots and automatic synonym generation for data distribution.
- Server-side PL/SQL logic

The Server Generator creates 100% database implementations directly from the Repository.

Database Design

It is useful to examine briefly the reasons why we should consider "designing" an Oracle 7 database system. There are a number of important reasons:

- Performance - the initial design of a system can have tremendous impact on its final performance
- Integration - without some sort of design specification it becomes very difficult to for a team of developers working on an application to produce a common application, this is equally true when the system being build needs to be integrated into existing systems.
- Communication - if a design exercise is undertaken, the decisions in that process need to be communicated to all members of the team.
- Reuse - it may well be possible to reuse existing successful design solutions

Some development organizations are familiar with the concept of producing an information model for the business requirement they are developing as a database system. Some organizations represent that information model as an entity relationship model. We will not cover the principles and techniques of ER modeling, but how we can translate the information model into a physical database using Designer/2000.

Using the Database Design Wizard the Entity Relationship Model can be translated into a database design, there are a number of significant advantages to doing this;

- Save development time
- Achieve consistent results
- Avoid a great deal of repetitive work
- Maintain the database design.

In practical terms the Database Design Wizard performs the following tasks for a database designer automatically and without error or misinterpretation.

- Entities become tables
- Attributes become columns
- Unique identifiers UIDS become primary and unique key constraints
- Relationships become foreign key columns and constraints
- If an entity has several uids, the designer can choose the most appropriate for the primary key constraint, the rest become unique key constraints
- Sub type entities are implemented as multiple or single tables under designer control
- Many to many relationships are automatically resolved by the creation of intersection tables.

The Database Design Wizard produces a complete design and the definitions produced could be used to generate the SQL DDL scripts. Often though, the first cut database design has been achieved, the data diagrammer is used to refine the database design and there a number of reasons as to why designers may want to go through this exercise.

For example, convenience database objects to may need to be added to the database schema, i.e. views may need to be created to hide the complexity of the design, sequences allowing the generation of certain column values. The application may also require additional columns for auditing , journalling etc. There is also the whole area of performance, certain columns in the database will need to be indexed for performance, and the structure of certain tables may need to be changed to speed up access times.

Let's examine in more detail the area of table and column design. Using the Data Diagrammer the designer can use the column definition spread table to specify and modify column definitions, the benefit of doing this is so that we can utilize other Designer/2000 utilities as indicated in the table below.

Column Property	Description	DDL Server Generator	Database Sizing Utility
Sequence	The order in which the column appear in the table	✓	
Datatype	The datatype of the column	✓	✓
Avg Len	Estimate of average length		✓
Max Len	Column length	✓	✓
Dec Places	Accuracy of number columns	✓	✓
Optional	Whether the column will allow NULL values	✓	✓
Initial Vol	Estimated percentage		✓
Final Vol	Estimated percentage		✓
Default Value		✓	

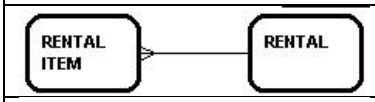
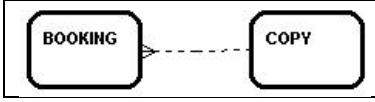
The Oracle 7 server allows for the definition of data integrity constraints that are automatically enforced by the server. The types of integrity constraint are :

- Nulls - i.e. mandatory or optional columns
- Unique Column Values - all values in a column are unique
- Primary Key Values - ensures that all rows in a table can be uniquely identified by the values in a column (or set of columns), the implication being that the column is mandatory and all of its values are unique.
- Referential Integrity - a rule defined for a column that allows inserts or updates only values that exist in another table. Also includes rules that dictate the type of data manipulation allowed on referenced data and what actions to be performed on referencing data as a result of such manipulation
- Complex Integrity Checking - user defined rule for a column that allows or disallows certain values.

The Data Diagrammer allows you to simply create and utilize the above constraints

- NULLS - simply define a column as mandatory by setting it's "Optional?" property to false
- Primary Keys - The Database Design Wizard automatically creates a primary key constraint for the source entity primary uid, or using the Data Diagrammer directly you can edit or create primary key constraints and components in the edit table dialog.
- Unique Keys - use the Data Diagrammer edit table dialog.
- Foreign Key - diagramatically or in edit table dialog.
- Foreign Key Delete and Update rules :
    - null - no delete/update rule
    - cascades - deletes/updates a row from the table if parent row is deleted/update
    - restricted - prevents deletion/update of parent rows
    - nullifies - updates the foreign key to null if the parent row is deleted/updated
    - defaults - updates the foreign key to a specified value if the parent row is updated.

Oracle 7 implemented✓

	DELETE	UPDATE
	CASCADE✓	CASCADE
	RESTRICT✓	RESTRICT✓
	NULLIFY or DEFAULT	NULLIFY or DEFAULT

As the above tables describes ONLY the Delete Cascade, Delete Restricted and the Update Restricted rules are enforced by the Oracle 7 server. The Forms Generator however, Designer/2000 can create the application code to support all delete and update rules at the application level.

- Check Constraints - are implemented using the Data Diagrammer edit table dialog.

Procedural server-side logic design

There are two areas of interest in writing server side procedural logic , PL/SQL stored procedures, packages and functions and database triggers. We will first examine the area of database triggers.

Database triggers can be used to supplement declarative integrity constraints to enforce business rules. Because the triggers are written in PL/SQL the rules can be completely customized. Triggers can be commonly used for

- Implement referential integrity rules not supported by declarative constraints
- Enforce complex business rules
- Automatically generate derived column values
- Provide transparent event logging and auditing
- Enforce complex security authorizations

The Module Logic Navigator provides a graphical environment to aid creating and maintaining both free standing PL/SQL modules and database triggers.

The Module Logic Navigator provides a selection tree and an edit window to define the PL/SQL trigger module.

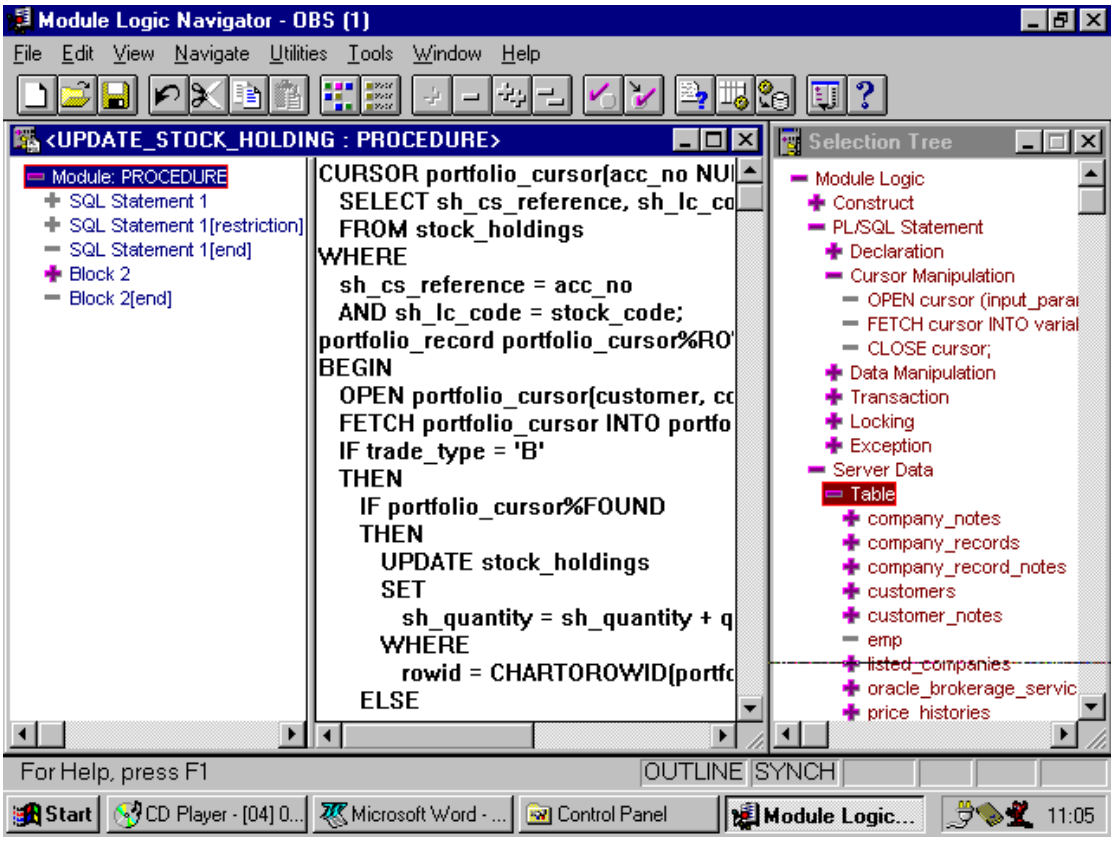


Figure 2 - Module Logic Navigator

Selection Tree

The Selection Tree provides correct PL/SQL constructs and allowable data values that can be dragged and dropped to the editor or the outliner.

Edit Window

The edit window is divided into two sections

- Outliner - shows the module structure in an abbreviated form
- Text editor - the text editor shows the full text of the module, it also automatically formats any PL/SQL. If you have existing PL/SQL modules defined in text script files, these can be imported directly into the Module Logic Navigator

The key benefits of using the Module Logic Navigator are its associated utilities, the Syntax Checker and the Generate Summary Usages utilities. The Syntax Checker provides a mechanism for checking the syntax of the PL/SQL code before it is implemented on the server, and the generate summary usages produces the table usages of the module. This then allows the PL/SQL module to be cross referenced in the any of the impact analysis tools in Designer/2000. For example, using the Matrix Diagrammer database designers can automatically see the impact of changing a database structure i.e. table/column on the modules in the system.

The method for creating for free standing PL/SQL server side procedures is exactly the same as described for database triggers., and all of the benefits outlined for database triggers apply equally to free standing server side PL/SQL packages, procedures and function modules.

Server Generation

The server generator is used after the designer has finished the database design, the generator will produce DDL command files that can be used to build a “live” database. The files generated are SQL*Plus command files and are therefore portable to any Oracle environment Below is a table that describes all of the database objects that the server generator will produce scripts for.

Oracle 7 Object Name	Generate Y/N
Table	Y
View	Y
Synonym	Y
Index	Y
Sequence	Y
Cluster	Y
Integrity Constraints	Y
including Primary Key	Y
Foreign Key	Y
Check Clause	Y
Default Value	Y
Null	Y
Snapshots	Y
Rollback Segments	Y
Tablespaces	Y
Database	Y
Database Trigger	Y
PL/SQL Package	Y
PL/SQL Procedure	Y
PL/SQL Function	Y
Database Link	Y
Role	Y
User	Y
Grants	Y

In the previous sections we have seen how the Data Diagrammer and Module Logic Navigator can aid database design. The other significant tool that is extremely powerful is the Repository Object Navigator. This tool supports the designer in allowing him/her to manipulate repository objects extremely easily. For example, a designer can create a space definition in the repository, this space definition can be applied to multiple database objects simply by selecting the required objects and changing the space definition property to the name of your newly created space definition. This means that whenever those objects are generated they will automatically be generated with the correct space definition. This reuse capability is available in many parts of the Designer/2000 product set and significantly aids in the speed and quality of the generated database structures.

In Designer/2000, there are a number of pre-defined quality reports provided. Their purpose is to check the completeness of the element details defined in the repository, and these are typically run before any generated SQL DDL scripts are generated and run.

Quality Reports

- Table Quality Control
- Database Trigger Quality Control
- Groups/Users Quality Control
- Package/Procedure Quality Control

The area of quality also includes the subject of documentation, not only does the Designer/2000 product provide an excellent repository of documentation, the SQL DDL script files are fully commented, thus allowing subsequent users understand their functionality. Figure 3) below is sample output from the Server Generator.

```
REM
REM  This ORACLE7 command file was generated by Oracle Server Generator
REM  Version 5.5.7.0.0 on 26-OCT-95
REM
REM For application OBS version 1 database OBS7
REM
REM TABLE
REM      CUSTOMERS

REM
PROMPT
PROMPT Creating Table CUSTOMERS
CREATE TABLE customers(
 cs_reference                NUMBER(6,0)              NOT NULL,
 cs_address_line_1           VARCHAR2(40)               NOT NULL,
 cs_last_name                VARCHAR2(30)               NOT NULL,
 cs_first_name               VARCHAR2(30)               NOT NULL,
 cs_telephone_number              VARCHAR2(15)                NOT NULL,
 cs_address_line_4             VARCHAR2(40)               NULL,
 cs_address_line_2             VARCHAR2(40)               NULL,
 cs_address_line_3             VARCHAR2(40)               NULL,
 cs_lc_code                  VARCHAR2(4)              NULL,
 cs_acc_balance                NUMBER(11,2)               NULL,
 cs_od_allowed               VARCHAR2(1)              DEFAULT 'N'  NOT NULL
              CHECK ( cs_od_allowed IN ( 'Y' , 'N' )  )
)
STORAGE  (
  INITIAL   777
  NEXT   11
  MINEXTENTS  1
  MAXEXTENTS  99
  )
;

COMMENT ON TABLE customers
   IS 'Created from Entity CUSTOMER by OBS on 24-AUG-94';

COMMENT ON COLUMN customers.cs_reference
   IS 'A unique system-generated id for this customer';

COMMENT ON COLUMN customers.cs_address_line_1
   IS 'First address line';

COMMENT ON COLUMN customers.cs_last_name
   IS 'Last Name of Customer';

COMMENT ON COLUMN customers.cs_first_name
   IS 'First name of Customer';

COMMENT ON COLUMN customers.cs_telephone_number
   IS 'Telephone area code and number';

COMMENT ON COLUMN customers.cs_address_line_4
   IS 'Fourth address line';

COMMENT ON COLUMN customers.cs_address_line_2
   IS 'Second address line';

COMMENT ON COLUMN customers.cs_address_line_3
   IS 'Third address line';

COMMENT ON COLUMN customers.cs_lc_code
   IS 'A 4 character code to uniquely identify the Company.';
```

The SQL DDL script files that are generated are defined by types, i.e. the table creation is performed initially, and then any constraints that are required are applied by a separate script file, similarly with

index files etc. This gives the Designer and DBA a lot of flexibility in how they build the final database.

Maintenance

Designer/2000 benefits the maintenance stage of an Oracle 7 based system as well in the construction of that system.

The maintenance stage of an application is typically where a large amount of effort is expended in the overall life cycle of that application, and that is why it is important to have a clear understanding of the application and all its dependencies. The use of the Designer/2000 repository can help dramatically in a number of ways in understanding dependencies

1. Objects have relationships and these can be clearly documented and understood by using the repository

2. Objects need to be documented for future reference and understanding

3. Typically there will be iterations and new versions of the application over a period of time, and it is important to be able to move the application forward with those changes.

Let us take each of those points in turn:

1. The Designer/2000 repository allows the designer to specify the relationships between all of the possible database objects

- Tables and columns
- Table and column constraints - including foreign key constraints
- Tables and Indexes
- Tables and Views
- Tables and Database Triggers written in PL/SQL
- Tables and PL/SQL stored procedures etc.
- Tables and Users/Roles
- Tables and Tablespaces

Having this ability to specify relationships means that a database designer can clearly understand what the impact of making changes on the system will be. Understanding the impact of change is half the battle of managing that change, rather than making a change and having no real idea as to what the impact will be. Designer/2000 provides a very strong capability in the area of impact analysis, either by the use of printed standard reports documenting the relationships or by the use of the Matrix Diagrammer.

The Matrix Diagrammer is a tool for cross-referencing, creating and manipulating Repository elements in the form of matrices. A matrix diagram consists of two axes, each displaying a different element type, and intersection cells, which indicate whether a relationship or association exists between two elements. Typically, an intersection element shows how one element uses or applies to another.

The Designer/2000 Matrix Diagrammer provides:

- Facilities for cross-referencing elements in the Repository
- Ability to view and update elements held in the multi-user Repository
- Ability to create new elements in the Repository
- Settings to control the appearance of the matrix, such as the font and justification for element properties
- Settings for filtering and automatically ordering elements on a matrix
- Choice of three viewing modes: standard, iconic and micromap

Figure 4) below shows an example of a matrix of tables against modules.

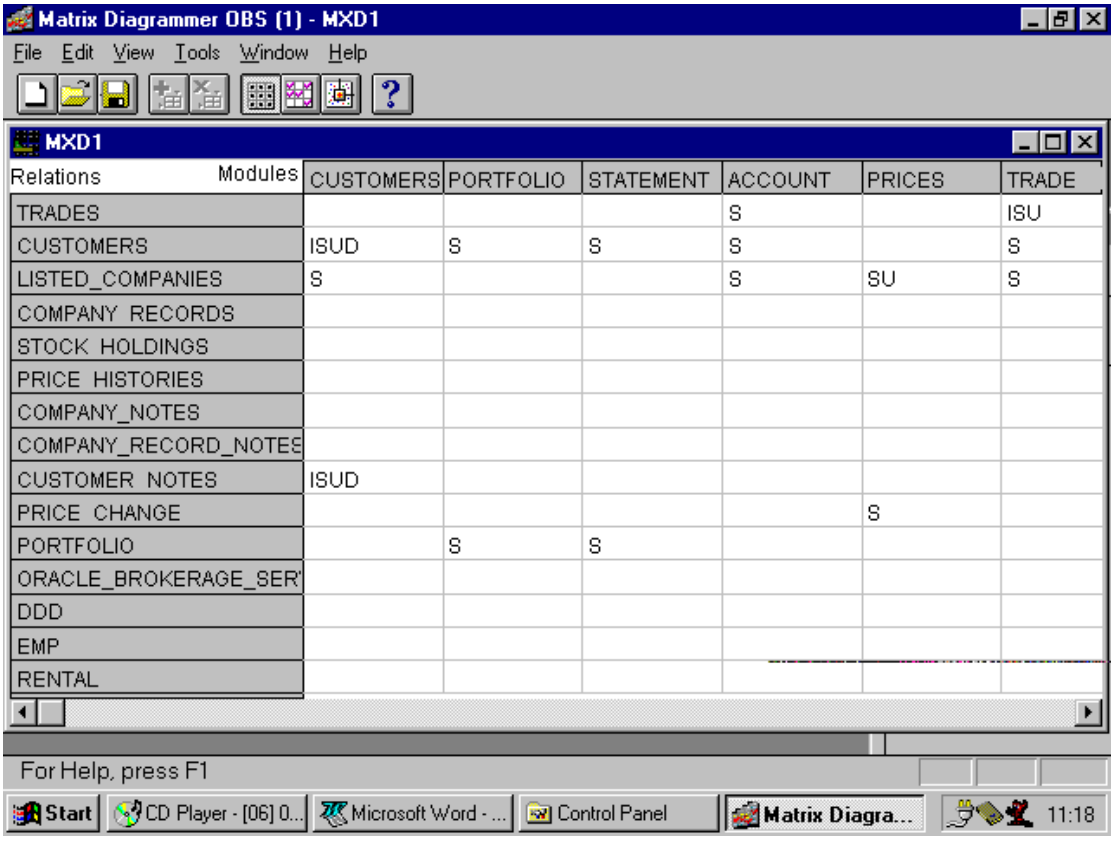


Figure 4 - Matrix Diagrammer

The above figure illustrates an extremely useful of a matrix example of all tables in a database against all PL/SQL modules in a database, thus allowing to graphically see which modules would be potentially affected by the change of a table.

2. The Designer/2000 repository provides the facility to include documentation for the application system.

Firstly by the use of Application Systems; an Application System allows designers to gather together all the repository elements , i.e. tables, modules etc. that make up an application into one unit and manage that unit. An application system can have different states, i.e. frozen, and it also may have many versions. It also serves as the basis for access control, and elements can be shared across application systems

3. Applications evolve and change over a period of time, and Designer/2000 provides a number of extremely useful facilities to help designers move their applications forward in a controlled manner.

The Database Design Wizard can be run in an iterative manner, so if the entity model requires changes, those changes can be reflected down into the physical database design by re-running the Database Design Wizard. Those changes can then be reflected into the live database by rerunning the Server Generator and creating SQL ALTER script files for the database objects that need modification. If this approach is combined with the use of the reconciliation utility, i.e. a utility that generates a report that lists live database objects side-by-side with Repository definitions, and reflects differences that may exist between the two; the Database Designer has a very powerful set of tools for managing an ongoing Oracle 7 development.

Design Recovery

Many existing systems have little or no formal documentation but typically manage valuable data about a business. These are often described as “legacy systems”. They must be maintained but the information to do so effectively is missing, the maintenance programmers need to know.

- Where the data is being stored
- What format the data is in
- How the data is used by the existing system
- How the application system fits in with the current business practice

Eliciting this information is know as Design Recovery.

- Deriving the database definitions and module data usages (i.e., the reverse engineered definitions) from the real-world system
- Deriving the existing business model from the reverse engineered data
- Analyzing the new business requirement and comparing it with that derived from the existing system
- Redesigning and implementing the new system, if required, to satisfy the new business requirement

The Reverse Engineer Database utility and Table to Entity Retrofit utility automates this reverse engineering process. Repository object definitions can be derived form the elements of the existing system if the system was developed on an Oracle 7 database. When reverse engineering operations have been performed, the new application system definition or definitions can be used as the basis for system development.

Stages Involved in Reverse Engineering

A reverse engineering operation can involve the following stages.

Reverse Engineering - Stage One

This stage covers reverse engineering database objects, creating module data usages of existing PL/SQL modules.

Reverse Engineering - Stage Two

This stage covers further steps required for recreating a business model, rationalizing the top-down business model with the reverse engineered data and completing the cross-referencing.

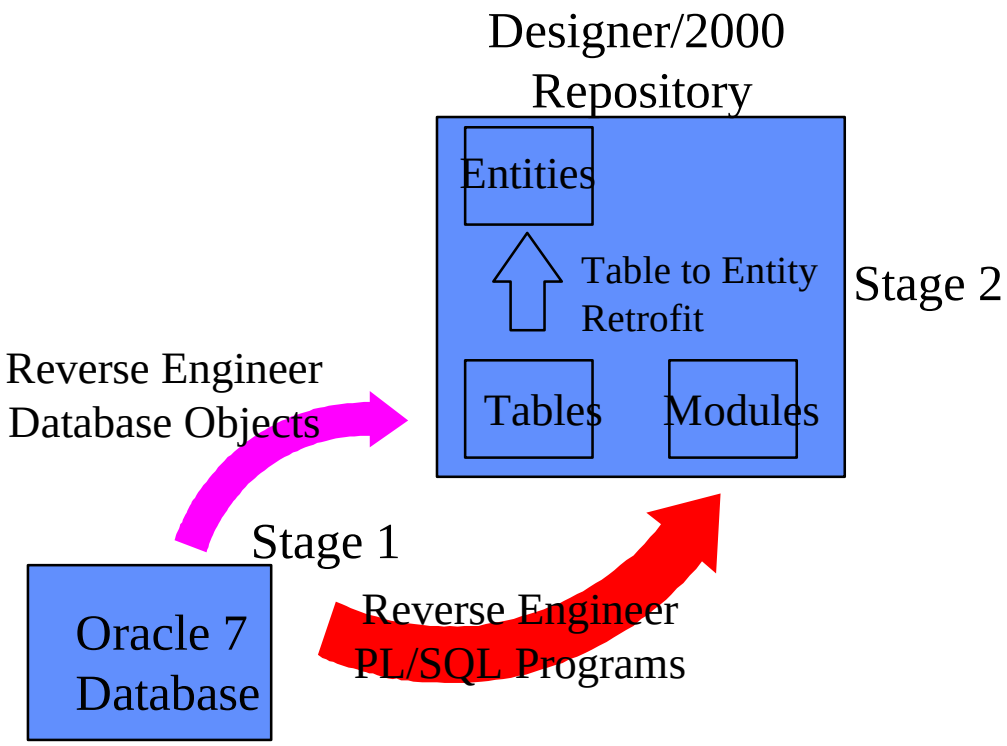


Figure 5 - Design Recovery

Reverse Engineering - Stage One

This stage can include:

- Reverse engineering database objects
- Creating module data usages

When this stage is complete most of the reverse engineering process has been accomplished. The application system, which until this time may not have been controlled nor documented, can now be maintained.

Reverse Engineer Database Objects

Reverse engineering database objects involves using the Reverse Engineer Database utility to load one or more definitions of an existing Oracle on-line database into the Repository. The reverse engineered repository object definitions should then be analyzed to identify and isolate:

- Primary, unique and foreign keys
- Sub-type and super-type entity implementations within, or across, one or more tables
- Denormalized data
- Duplicated data

Note: If primary, foreign and unique keys are presenin the Oracle Data Dictionary, they will have been automatically identified. Below is a table that details the database objects that can be reversed engineered.

Oracle 7 Object Name	Reverse Engineer Y/N
Table	Y
View	Y
Synonym	Y
Index	Y
Sequence	Y
Cluster	Y
Integrity Constraints	Y
including Primary Key	Y
Foreign Key	Y
Check Clause	Y
Default Value	Y
Null	Y
Snapshots	Y
Rollback Segments	N
Tablespaces	Y
Database	N
Database Trigger	Y
PL/SQL Package	Y
PL/SQL Procedure	Y
PL/SQL Function	Y
Database Link	N
Role	N
User	N
Grants	N

The Repository can now be used to define and implement changes and enhancements to the existing database. If definitions of new tables or other objects are created in the Repository they can be implemented in the existing Oracle Data Dictionary using the Generate DDL utility. If changes to existing repository objects (e.g., adding a new column to a table, add a new foreign key) are defined in the Repository they can be implemented in the Oracle Data Dictionary using the Alter Database Command Generator utility. If the Repository and Oracle Data Dictionary are out of step during maintenance, the Cross Reference utility can be used to indicate the differences, allowing the appropriate steps to be identified and enacted.

Creating Module Data Usages

Associations need to be defined between PL/SQL modules and tables/views/snapshots/columns to begin the cross-reference documentation for the implementation stage. This is achieved by using the create data usages utility

Reverse Engineering - Stage Two

This stage is optional, but it will provide a detailed account of the business model.

A normalized data model can be derived from the reverse engineered definitions of the existing business system.
The Table to Entity Retrofit utility should be used to derive a 'first-cut' entity model from the physical database definition tables.

The utility generates entities, attributes, relationships and unique identifiers from the following:

- Entity definitions are created from table definitions
- Attribute definitions are created from any columns not involved in foreign key constraints

- Relationships are created from the foreign keys of the specified table and the tables to which these foreign keys reference
- Primary keys are reverse engineered to primary unique identifiers
- Unique key constraints are reverse engineered to secondary unique identifiers

The repository object definitions created from this process should be examined carefully. For example, tables used for reference purposes may define domains with sets of known values, 'Type' columns may define entity sub-types, etc.

Rationalizing the Top-down Business Model with the Reverse Engineered Data

The work carried out in the previous stage in recreating a business model can be compared with the results of the Repository Reverse Engineer Database utility.

Completing the Cross-Referencing

Finally, you can set up the cross-references between the new business level data and the existing system implementation data. You can use the Repository Object Navigator to specify which new entities are implemented by which tables and which functions are implemented by which modules. The end product is a completely cross-referenced and fully integrated application system that gives you a firm basis for future enhancements.