

**CALIFORNIA STATE UNIVERSITY, SACRAMENTO**  
College of Business Administration

MIS 211 - Information Systems II


**Relational Algebra and SQL Homework Assignment**

Points: 45

Due: Wednesday, April 17

The objectives of this assignment are to relate the relational algebra presented in class and your textbook to basic SQL queries, and to familiarize you with Oracle8i/9i. It involves creating a user and two (database) tables, loading them with data from *control files*, importing a third table, and performing 14 queries on them. Follow the ScreenCam presentations for logging into SQL Plus, creating and importing the tables, and building SQL queries. You will need to capture your session in a spool file to receive credit (see class notes).

**Suggestion.** Plan ahead and read chapters 12 and 13 before attempting this assignment to avoid wasting many hours.

 **Note.** Windows XP users MUST install Oracle9i; 8i NT will NOT work. Windows 2000 users have a choice between Oracle9i and 8i NT. Install Oracle8i Windows 98 if you have Windows 98 and ME installed. If you are a Windows ME user, be sure to refer to the bulletin regarding the modification to the win.ini file that MUST be completed BEFORE Oracle8i is installed.

### Creating a User

Create user MIS211 with an assigned password of MIS211 (or any other user and password) in Oracle Navigator (Windows 98 users). Be sure to grant your user a DBA role so you may create and import tables.

### Windows NT, 2000 and XP Users

Unfortunately, Oracle does not install a DBMS Navigator with its Windows NT, 2000 and XP versions. Users must be created in SQL Plus with the CREATE USER and GRANT SQL commands. To initially open SQL Plus, use user name SYS and password CHANGE\_ON\_INSTALL (or SYSTEM and MANAGER). The syntax for creating a user is as follows:

```
CREATE USER username IDENTIFIED BY password;
```

Grant your user a DBA role so you may create and import tables:



```
GRANT dba TO username;
```

The DBA status will allow you to create tables and views, and import tables. Use your user name and password for subsequent log ons.

### Defining the Tables

Create two tables, COMP\_PRODUCTS and COMP\_MANUFACTURERS. Use Oracle8i Navigator to define the COMP\_MANUFACTURERS table (NT users will have create this table in SQL Plus). Define COMP\_PRODUCTS in SQL Plus using the (SQL) CREATE TABLE command. The following tables list the column names, data types, lengths and constraints. Since data will be loaded into the tables from control files (category.ctl, comp\_products.ctl), the column names must be identical to those listed below. Otherwise, you will have problems loading data into them.

#### Comp Manufacturers

Column Name	Data Type	Length	Integrity Constraint
Manufacturer_code	varchar2	3	primary key
Manufacturer_name	varchar2	20	not null

#### Comp Products

Column Name	Data Type	Length	Integrity Constraints
Model_number	varchar2	15	primary key
Product_description	varchar2	65	not null
Product_line	char	1	foreign key (Product_lines)
List_price	float*		default 0
Retail_price	float*		default 0
Manufacturer_code	varchar2	3	foreign key (Comp_Manufacturers)
Stock_on_hand	int <sup>†</sup>		default 0
Stock_on_order	int <sup>†</sup>		default 0
Last_received_date	date		not null



\*floating point (decimal)  
 †integer

## Importing a Table

The third table, PRODUCT\_LINES, has been exported to a dump (dmp) file, H5.DMP. Use the IMP utility to import this table into your database. Both the table's structure and data will be inserted to the database. Thus, the table's definition nor data are of concern.

Create a DOS window and change to the directory of the dump file. The general syntax for using Importer is (this would entered at the DOS prompt):

```
imp user-name/password
```

where:        *user-name* and *password*        User name and password assign to the user (e.g., mis211). Be sure to separate the name and password with the “/”.

**Note.** This dump file cannot be imported into Oracle8 or any lower version. It is only compatible with Oracle8i and 9i.

View the table's definition in Oracle8i Navigator (properties) or through the DESCRIBE command in SQL Plus:

```
describe table-name
```

## Loading Data into the Tables

Once the tables have been defined, they can be loaded with data from two control files, Comps.ctl, and CompManf.ctl (both are contained in h5.exe) via SQL Loader (SQLLDR). Minimize the SQL Plus window (do NOT exit SQL Plus) and create a DOS window. Change to the drive/directory under which you have placed these files.

The general syntax to enter at the DOS prompt is as follows:

```
sqlldr user-name/password file-name
```

where:        *user-name* and *password*        User name and password assign to the user (e.g., mis211). Be sure to separate the name and password with the “/”.



*file-name*

The name of the control file (either Comps or CompManf).

Do this for both files. Before exiting the DOS window (i.e., enter EXIT and press the ENTER key) and returning to SQL Plus, examine the *log* files (music.log, comp\_products.log) to see if all records have been loaded into the tables. If they have not, resolve all conflicts before reloading the table. In most cases, the problem will stem from a misspelled column name or an incorrectly specified column width. To view the records that were not loaded, open the *bad* file (comps.bad, compmanf.bad) in a text editor.

**Note.** Before attempting this, download and view the ScreenCam presentation.

## Queries

Now that all tables contain data, you can perform the following queries. The details of the queries (i.e., predicate, column names) are of your own doing. Thus, it would be uncanny for two assignments to contain identical entries. Also, be creative! Do not follow the class notes or Screen Cam demonstrations too close. Doing so may be a detriment to your grade.

**Note.** When designing your queries, be sure they convey **usable information**. Queries that are comprised of a nonsensical assembly of columns and/or rows will not awarded full points. For example, constructing a query that only lists prices or retrieves no rows is not useful information. Please apply your *common sense!* *This isn't just an exercise, it's an investment in your future!*

***Carefully read the relational algebra!***

1. Selection. Perform a selection on the comp\_product table. The number of rows returned by the query should not exceed 15.

$$\sigma_{\text{predicate}}(\text{comp\_products})$$

2. Projection and Selection. Perform a project on your selection in (1).

$$\Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products}))$$

3. Projection and Selection using a logical AND and/or a logical OR. Add a logical AND and/or OR to your predicate in (2). The number of rows returned by the query should not exceed 10.

$$\Pi_{\text{column-list}}(\sigma_{\text{condition1} \{AND / OR\} \text{condition2}}(\text{comp\_products}))$$

4. Equi-join. Expand your query in (2) and perform an equi-join with the product\_lines or comp\_manufacturers table.



$$\begin{aligned} & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=product\_lines.column-name}} \Pi_{\text{column-list}}(\text{product\_lines}) \\ & \text{or} \\ & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=comp\_manufacturers.column-name}} \Pi_{\text{column-list}}(\text{comp\_manufacturers}) \end{aligned}$$

**Note.**  $\bowtie$  = equi-join

5. Equi-Join and Range. Perform another join between the comp\_products table and either the comp\_manufacturers or product\_lines table. Incorporate into your predicate a condition using the BETWEEN clause. The number of rows returned by the query should not exceed 15.

$$\begin{aligned} & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=product\_lines.column-name}} \Pi_{\text{column-list}}(\text{product\_lines}) \\ & \text{or} \\ & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=comp\_manufacturers.column-name}} \Pi_{\text{column-list}}(\text{comp\_manufacturers}) \end{aligned}$$

6. Equi-Join and Pattern Matching. Replace the BETWEEN in query (5) with the LIKE clause and the UPPER/LOWER function in the predicate. The number of rows returned by the query should not exceed 15.

$$\begin{aligned} & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=product\_lines.column-name}} \Pi_{\text{column-list}}(\text{product\_lines}) \\ & \text{or} \\ & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=comp\_manufacturers.column-name}} \Pi_{\text{column-list}}(\text{comp\_manufacturers}) \end{aligned}$$

**Note.** Pattern matching is performed on character strings (data), not numeric data. Performing a pattern match on numeric data is inappropriate.

7. Aggregation, GROUP BY / HAVING and Equi-Join. Perform a query using the COUNT, SUM, AVG, MAX and MIN aggregation functions (all 5). Apply them over selected groups of entities (i.e., rows) that share a common attribute value. Also, apply a format mask to all numeric columns (see pages 70-73 of the Enhanced Guide to Oracle8i textbook or refer to the Oracle online documentation). Include a comma (,) between the thousands and hundreds places (i.e., 9,999). For monetary values, include the dollar sign (\$) and decimal point. The number of rows returned by the query should exceed two (i.e., three or more) and should not exceed 15.

$$\begin{aligned} & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=product\_lines.column-name}} \Pi_{\text{column-list}}(\text{product\_lines}) \end{aligned}$$



$$\begin{aligned} & \text{or} \\ & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=comp\_manufacturers.column-name}} \Pi_{\text{column-list}}(\text{comp\_manufacturers}) \end{aligned}$$

8. Calculation and SYSDATE. Perform a query involving a calculation with SYSDATE (system date) and either the date\_released or active\_date column (of comp\_products) to determine the number of years and months between the two dates. Apply a date format mask to the date column you have selected and display its retained value (see page 31 of the [A Guide to Oracle8](#) textbook or refer to the Oracle online documentation); do NOT use the default mask. Attach an alias to the calculation and to\_char columns. Include other descriptive titles in your query. The number of rows returned by the query should exceed two (i.e., three or more) but be less than 16.

$$\begin{aligned} & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=product\_lines.column-name}} \Pi_{\text{column-list}}(\text{product\_lines}) \\ & \text{or} \\ & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\ & \bowtie_{\text{comp\_products.column-name=comp\_manufacturers.column-name}} \Pi_{\text{column-list}}(\text{comp\_manufacturers}) \end{aligned}$$

Refer to pages 94-98 of the [A Guide to Oracle8](#) textbook for arithmetic operations on a date data type. You may have to use the TRUNC (prevents rounding) and MOD (modulus; retains the remainder in division) functions. The syntax for the two are as follows:

TRUNC ( *arithmetic-operation* , *decimal-places* )

where:            *arithmetic-operation*            Composition of arithmetic operations (e.g., retail - cost / cost) or an aggregation function (e.g., avg(retail\_price)).

*decimal-places*            Number of decimals to be displayed. This can be any positive integer or zero. When not included, zero is assumed.

MOD ( *numerator* , *denominator* )

9. View. Create two union compatible views using projections on selections and equi-joins between the comp\_products and either the product\_lines or comp\_manufacturers tables. The number of columns specified in the projections for each view may vary. However, they (projections) should include at least two common columns (i.e., union compatible). When designing the selections, allow for an overlap of entities between the views. Thus, some entities will appear in both views.



$$\begin{aligned}
 & \text{view-name} = \\
 & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\
 & \times_{\text{comp\_products.column-name=product\_lines.column-name}} \Pi_{\text{column-list}}(\text{product\_lines}) \\
 & \text{or} \\
 & \Pi_{\text{column-list}}(\sigma_{\text{predicate}}(\text{comp\_products})) \\
 & \times_{\text{comp\_products.column-name=comp\_manufacturers.column-name}} \Pi_{\text{column-list}}(\text{comp\_manufacturers})
 \end{aligned}$$

Display the contents (all columns and rows) of both views. The number of rows produced in the view should not exceed 10. Do NOT apply any conditions to limit the number of rows or restrictions on the number of columns displayed.

**Note.** Before creating the views, study the requirements of (10) through (12).

10. Union. Perform a union on the views in (9). The number of rows produced by the union should be less than the sum of the two views.

$$\Pi_{\text{column-list}}(\text{view1}) \cup \Pi_{\text{column-list}}(\text{view2})$$

11. Intersection. Find the computer products both views in (9) share through their intersection. There should be at least (i.e., a minimum of) 3 rows. Zero rows is not acceptable.

$$\Pi_{\text{column-list}}(\text{view1}) \cap \Pi_{\text{column-list}}(\text{view2})$$

12. Set Difference. Find computer products that are unique to one view. There should be at least (i.e., a minimum of) 3 rows.

$$\Pi_{\text{column-list}}(\text{view1}) - \Pi_{\text{column-list}}(\text{view2})$$

**Note.** The number of rows returned by the union in (10) plus the number returned by intersection in (11) should equal the sum of the rows in both views (i.e.,  $\text{view1} + \text{view2}$ ). The number of rows returned by the intersection in (11) plus the number returned by the set difference in (12) should be equal to the total number rows in  $\text{view1}$ .

13. ANY/ALL with a Subquery. Perform a subquery that incorporates either the ANY or ALL. Use either the `comp_products`, `product_lines`, `comp_manufacturers` tables and/or your views in (9). Explain the purpose of your subquery and the information it presents next to your output.

```

SELECT column-list-1 FROM {table-1 | view-1}
WHERE column-name-1 {relational operator} {ANY | ALL }
  ( SELECT column-list-2 FROM {table-2 | view-2}
    WHERE condition )

```



14. EXIST/NOT EXIST with a Subquery. Following the table/view specifications in (13), create a EXIST/NOT EXIST in your subquery. Explain the purpose of your subquery and the information it presents next to your output.

```
SELECT column-list-1 FROM {table-1 | view-1}
WHERE {EXISTS | NOT EXISTS}
      (SELECT {* | column-list-2} FROM {table-2 | view-2}
       WHERE condition)
```

## Tangibles

To receive credit for this assignment, submit a printed copy of your spool file in a 9 × 12-inch manilla envelope; a diskette is NOT required. The listing should include the CREATE TABLE for Comp\_products, and queries (1) through (14). Tab the pages with your queries, highlight the SQL command (with a color marker, preferable yellow), and write the number corresponding to the requirement next to the query. These indicate the *attempt* you want graded. As stated in the past, credit cannot be awarded for assumed work. **Unmarked work will not be graded.**

**Note.** Do not worry about errors in your spool file listing. This provides you with more evidence that the work is yours! Also, multiple sessions are acceptable.

Please also be aware of the *one assignment, one grade* rule. Any assignments that have an uncanny resemblance will be considered a violation of ethical class behavior. You may work with others, but the assignment you submit should be the product of your own work.

As stated in the syllabus, homework assignments are due at the beginning of class. Late assignments will NOT be accepted.

## Dropping Your Tables

If you are working on a public access computer (i.e., one in a lab), remove your tables from the database before exiting. Use the SQL DROP TABLE command followed by COMMIT to permanently delete the tables from the database.

```
DROP TABLE table-name;
COMMIT;
```

These tables will be issued to you for the next assignment.

