

Oracle Studio #1 -- Querying an Oracle Database

CSCI-4380 Database Systems

Fall 2001

1. Introduction

In this studio, you will begin to use the Oracle database system. Oracle is one of the most widely used relational database systems. It runs on a variety of platforms, and is able to handle very large databases (much larger than Microsoft Access can handle by itself). Here at Rensselaer, Oracle runs in client / server mode. One particular RCS UNIX workstation runs the Oracle server and contains all the databases that Oracle manages. Many of the Windows NT PC workstations on campus can function as Oracle clients that can access databases on the Oracle server. Oracle applications such as SQL*PLUS, the forms package, and the report generator run on the client workstations. When run, they use SQL*NET to communicate with the Oracle server to access a database. Currently, the NT workstations in Pitts 4114 and in VCC North have the Oracle client tools installed.

In this studio you are going to use SQL*PLUS to query an Oracle database. The database you will query is *The Movies Database* (the full database this time). You will have a chance to practice writing SQL queries as well as learning how to use SQL*PLUS. To make the studio more interesting, some of the queries that you write will be based on a game called *Six Degrees of Kevin Bacon*. If you are not familiar with this game, it is explained below.

2. Getting Started

To use SQL*PLUS, you must have an Oracle account for the Oracle server. You will get this account from your studio instructor. The account includes both a userid and a password.

Click on the Start button in the lower left corner of the screen. Then go to Programs in the menu that pops up. In the Programs menu look for a folder that includes Oracle as part of its name (there may be several). Look in these folders to find SQL Plus 8.0 and click on it. This will start SQL*PLUS.

SQL*PLUS begins by displaying a dialogue box that you use to log into the Oracle server. This dialogue box requires that you enter three things. The first is the userid for your Oracle account, the second is the password for your Oracle account, and the third is the host server name ora8. The name ora8 is the name of the RCS server for Oracle that you will be using. Once you have enter these values and clicked on the OK button, SQL*PLUS opens its main window and displays messages similar to the following:

```
SQL*Plus: Release 8.0.5.0.0 - Production on Wed Feb 16 9:10:20 2000
```

```
(c) Copyright 1998 Oracle Corporation. All rights reserved.
```

```
Connected to:
```

```
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
```

```
With the Partitioning and Java options
```

```
PL/SQL Release 8.1.5.0.0 - Production
```

```
SQL>
```

The "SQL>" prompt indicates that SQL*PLUS is ready for you to type your first command. All SQL commands should end with ";" in SQL*PLUS. SQL*PLUS is case insensitive for attribute and relation names and for SQL keywords like select, from, and where.

3. Changing Your Password and Default Editor

Before going any further, you would probably like to change the password for your Oracle account. You use the ALTER USER command to do this. Type the following command in the SQL*PLUS window after the command prompt (i.e., SQL>):

```
alter user <account> identified by <newpassword> ;
```

where <account> is the userid of your Oracle account (the same as your RCS id), and <newpassword> is the new password that you want to begin using with your account. Keep in mind that an Oracle password cannot start with a number. If the password is changed successfully, SQL*PLUS will display the message "User altered." Note that you must type a semicolon at the end of the line. If you forget to do this before you type the "Enter" key, SQL*PLUS will wait for you to type the semicolon on the next line before it processes the command.

You are not prompted to type your new password a second time to verify that you typed it correctly. So, be careful that you type it correctly the first time. Make sure that you remember what your new password is, because no one else can tell you what it is once you change it.

4. Six Degrees of Kevin Bacon

There is a party game called *Six Degrees of Kevin Bacon*. The premise of this game is that any actor is within six degrees of separation from Kevin Bacon. The goal of the game is to show that this is true for any given actor by finding a path from that actor to Kevin Bacon. This path can have no more than six degree of separation in it.

What is a degree of separation? A degree of separation is an appearance in a movie with another actor that is on a shorter path to Kevin Bacon. For example, Mathew Perry was in *A Night in the Life of Jimmy Reardon* with River Phoenix, who was in *Sneakers* with Robert Redford, who was in *Indecent Proposal* with Oliver Platt, who was in *Flatliners* with Kevin Bacon. Hence, for this path, Mathew Perry has three degree of separation from Kevin Bacon, River Phoenix has two degrees of separation, Robert Redford has one degree of separation, and Oliver Platt has zero degrees of separation.

As another example, Drew Barrymore was in *Poison Ivy* with Tom Skerritt, who was in *Top Gun* with Tom Cruise, who was in *A Few Good Men* with Kevin Bacon. Hence, for this path, Drew Barrymore has two degrees of separation from Kevin Bacon, Tom Skerritt has one degree of separation, and Tom Cruise has zero degrees of separation.

You might be interested to know that over 80% of the 14,001 actors in *The Movies Database* are separated from Kevin Bacon by three degrees or less.

5. Connecting to The Movies Database

The database owner has granted you read access to *The Movies Database*. Thus, you can access any of the relations in the database in an SQL query, but you cannot make any updates to these relations.

The movie database lies under the schema "movies." Consequently, you must append the prefix "movies." to the relation names. For example, to refer to the Films relation, use "movies.Films" or "movies.films" (Oracle is case insensitive).

You might find it simpler to use SQL aliases in the FROM clause of a query to avoid using this prefix

every time you refer to a relation in a query. For example, if a query involves the Films relation, then in the FROM clause for the query type "FROM movies.Films F". This defines "F" as an alias for "movies.Films" so that you only need type "F" in the rest of the query to refer to this relation.

6. Problem Statement

Now you are ready to query *The Movies Database*. We will start with an easy query.

6.1 The First Query

As your first query to *The Movies Database*, suppose that you want to find all the movies in which Kevin Bacon appears. If you look at the schema for the database, you will see that this information is in the Casts relation. However, you will also need to use the Actors relation to find the ActorId for Kevin Bacon and the Films relation to find the titles of the movies identified by FilmId in the Casts relation. Thus, the SQL query that you need is the following:

```
SELECT      movies.Films.Title
FROM        movies.Films, movies.Casts, movies.actors
WHERE       movies.actors.StageName = 'Kevin Bacon' AND
           movies.actors.ActorId = movies.Casts.ActorId AND
           movies.Casts.FilmId = movies.Films.FilmId ;
```

To enter this query, simply type it as it appears above after the "SQL>" prompt in the SQL*PLUS window. SQL*PLUS will number the lines of the query for you as you type. When you terminate the query with a semicolon, SQL*PLUS will attempt to parse and process the query. If you typed the query correctly, you should see the result displayed in the SQL*PLUS window. This result should include the following six films: *Footloose*, *Flatliners*, *JFK*, *The Big Picture*, *Murder in the First*, and *A Few Good Men*.

If you mistyped the query, you will instead see an error message or an incorrect result. In this case you must edit the query to correct it and then try again. Type "edit" at the "SQL>" prompt that immediately follows the error message or incorrect result. Note that there is no semicolon after "edit". This invokes the default editor to edit the query. This editor is a standard WYSIWYG editor. Click where you want to make a change. Use the Backspace or Delete keys to remove text. Type any new text that you want to add. Make the necessary corrections to the query, and then save and exit the editor using the File menu at the top of the editor window. Before you save the query make sure that it does not have a semicolon at the end. While a semicolon at the end is necessary when you type a query the first time, SQL*PLUS does not want a semicolon at the end of a query after it has been edited.

After saving the query and exiting the editor, a new "SQL>" prompt is displayed by SQL*PLUS. Type "run" after this prompt to get SQL*PLUS to run the new version of the query that you just finished editing. Again, there is no semicolon after "run". You can repeat this process as often as necessary to get the query to run successfully and produce the correct result.

An alternate form for the query above that involves less typing is the following:

```
SELECT      F.Title
FROM        movies.Films F, movies.Casts C, movies.actors A
WHERE       A.StageName = 'Kevin Bacon' AND
           A.ActorId = C.ActorId AND
           C.FilmId = F.FilmId ;
```

This version of the query makes use of aliases in SQL to avoid typing the database schema name at the front of every relation name in the query.

6.2 Actors with Zero Degrees of Separation from Kevin Bacon

Before starting this query, type the following command in the SQL*PLUS window at the SQL> prompt:

```
Column StageName format A35;
```

This command will be explained later. It improves the formatting of the results of the queries that you are about to write.

The next query you should try is a query to find all the actors with zero degrees of separation from Kevin Bacon. These are the actors who have appeared in a movie with Kevin Bacon. The first thing that you must do is design the SQL query needed to do this. This SQL query is a bit tricky. It requires that you use aliases in SQL for the Actors and Casts relations. The reason is that your query must be able to refer to two different tuples in the Actors relation simultaneously -- the tuple for Kevin Bacon and the tuple for another actor that appears in a Kevin Bacon movie. You must also be able to refer to two different tuples in the Casts relation simultaneously -- the tuple that identifies Kevin Bacon as an actor in a movie and the tuple for another actor in that movie. If you avoid using nested subqueries, your query will execute much faster.

Once you have designed your SQL query, enter it into SQL*PLUS and see if it works. If it doesn't work, you will need to edit it and try again.

How many actors in *The Movie Database* have zero degrees of separation from Kevin Bacon? If you eliminate Kevin Bacon from the answer and eliminate duplicates, then there should be 44 actors in your result relation. One of the actors is listed as "sa" which means "some actor" whose identity is unknown. If you don't eliminate duplicates and Kevin Bacon from your answer, then there should be 53 tuples in your result relation.

6.3 Actors with One Degree of Separation from Kevin Bacon

Now suppose that you want to find all the actors who are separated from Kevin Bacon by one degree of separation. That is, all actors who appear in a film with an actor that appears in a Kevin Bacon film. You can retrieve the stage names of all such actors using a single SQL query. This query extends the query you wrote above for finding actors with zero degrees of separation from Kevin Bacon. In this case, however, you will need additional aliases of the Casts and/or Actors relations. Again, try to avoid using nested subqueries. Your query will run in less time if you do.

See if you can successfully write and run the query for finding all the actors with one degree of separation from Kevin Bacon. The query result should include tuples with two attributes. The first is an actor with zero degrees separation from Kevin Bacon. The second is an actor with one degree of separation from Kevin Bacon. The path from Kevin Bacon to the second actor goes through the first actor. For example, one of the tuples in the result should be:

STAGENAME	STAGENAME
-----	-----
Oliver Platt	Robert Redford

since Robert Redford was in *Indecent Proposal* with Oliver Platt, who was in *Flatliners* with Kevin Bacon.

If you eliminate duplicate tuples in the answer, make sure that you don't use either Kevin Bacon or "sa" as either the first or second actor in a result tuple, and make sure that the second film is not the same as the

first film on the path, then there should be 1786 tuples in the result of your query.

You can of course continue extending the query to compute higher degrees of separation from Kevin Bacon. For example, if you compute two degrees of separation, you will find 88,471 tuples in the result. However, executing this query requires considerable time both to compute the answer and to display it. Doing so will seriously impact the performance of the Oracle Server for other students trying to use it. Therefore, please do not compute degrees of separation beyond one.

6.4 Creating a New Relation

Since continuing the game to higher degrees of separation is not practical, we will instead look at creating a new relation in an Oracle database. Suppose that you want to extend *The Movies Database* to include TV shows. You can do this by adding a new relation to the database. This new relation will be called TVShows and will have the following attributes:

TVId	integer (required, primary key)
Title	string (length 25, required)
StartYear	integer (required)
EndYear	integer
Network	string (length 3)
Episodes	integer
ShowType	string (length 10)

To create this relation in Oracle, you must use the CREATE TABLE command in SQL. Remember that the syntax for this command is the following:

```
CREATE TABLE <name>
(
    attr1    type [ NOT NULL ],
    attr2    type [ NOT NULL ],
    . . .
    PRIMARY KEY (attri, attrj, . . .),
    FOREIGN KEY (attrk) REFERENCES <relation>(<attribute>)
    . . . );
```

In this relation, the primary key is the TVId attribute and there are no foreign keys. In Oracle, "integer" is a valid domain type. Strings are represented in Oracle using the "varchar2(length)" domain type.

You now have everything that you need to create this new relation in Oracle. Type the necessary SQL command in the SQL*PLUS window to do this. If you get an error message, fix the command and try again. If the relation is created incorrectly, delete the relation by using the DROP TABLE tablename; command and try again.

6.5 Adding Data to a Relation

Once you have created the new TVShows relation, you can put data into it. To add tuples to a relation, you use the INSERT command in SQL. The general form for this command is the following:

```
INSERT INTO <relation> [ (attr1, attr2, . . .) ]
VALUES (<list of values>);
```

If you want to omit the value for one or more attributes when inserting a new tuple, you can either specify NULL as the value for those attributes, or list only those attributes with values in parentheses after the relation name. In the latter case, the VALUES clause contains values for only the listed attributes.

Use the INSERT command to enter the following TV shows into your new relation:

ID: 4077
Title: 'M*A*S*H'
Start Date: 1972
End Date: 1983
Network: 'CBS'
Episodes: 251
Show Type: 'comedy'

ID: 999
Title: 'Soap'
Start Date: 1977
End Date: 1981
Network: 'ABC'
Episodes: 92
Show Type: 'comedy'

ID: 567
Title: 'E.R.'
Start Date: 1994
End Date:
Network: 'NBC'
Episodes: 85
Show Type: 'drama'

ID: 123
Title: 'The X-Files'
Start Date: 1993
End Date:
Network: 'FOX'
Episodes:
Show Type: 'scific'

Now you might want to query the TVShows relation to see that the data was entered correctly.

6.6 Formatting the Output of a Query

By this time you have probably noticed that the default formatting for the output of a query often leaves much to be desired. Particularly if a tuple requires more than one line to display it, the screen can become difficult to read.

To solve this problem, you can use the COLUMN command in SQL*PLUS to specify how to format each attribute in the result of a query. For example, if you want to reduce the number of characters used to display the Title attribute from the TVShows relation when it appears in the result of an SQL query, you can enter the following command:

```
column Title format A12;
```

This command says that the Title attribute is to be displayed in a field 12 characters wide. If a title is longer than 12 characters, it is wrapped onto multiple lines within the 12-character field.

The general form of the column command is the following:

```
column <attribute> <format options> ;
```

Once a command is executed for a particular attribute, the format option stays in effect for the rest of the SQL*PLUS session, unless you change it with another column command. Some of the format options that can be specified include:

format An	format a varchar2 value as a field of width n
truncate	truncate varchar2 values after one line in the display field
clear	reset the formatting to the default
fold_after	start a new line after the field for this attribute

heading 'text' display text as the heading for this attribute

Experiment with the column command using the TVShows relation to see how the different format options work. For example you might want to truncate TV show titles to 10 characters when they are displayed. Or you might want to change the heading for the ShowType attribute to something like "Category".

6.7 Modifying Data in a Database

Suppose now that you want to modify some of the data in your new relation. For example, maybe you want to add to the tuple for *The X-Files* that there are 112 episodes. To do this, you use the update command in SQL. The general form of the update command is the following:

```
UPDATE <relation>
SET <attribute> = <expression>
WHERE <condition>
```

Make the above modification to the database and then query the TVShows relation to see that the change was made correctly.

6. Things to Think About

SQL*PLUS is a general and flexible interface to Oracle, but it is not particularly user friendly. How would you improve this? Can you do it without sacrificing the flexibility and generality of the interface?

You might try computing higher degrees of separation from Kevin Bacon using the Microsoft Access version of the database. The Access version of the database can be used to compute degrees of separation zero, one, and two using single SQL queries. (Higher degrees may be possible, but this has not been tested). The Microsoft Access version of the database is located in the course web site.

If you are interested in exploring the game *Degrees of Kevin Bacon* in more detail, you might find the following web site of interest: <http://www.cs.virginia.edu/~bct7m/bacon.html> . For a web-based movies database see the following web site: <http://www.imdb.com/> .

In the next studio you will learn how to use embedded SQL with Oracle. The following studio will look at the forms package for Oracle that allows you to build an application around a database much like you did with Microsoft Access.

7. If you finished the studio early try this optional part

If you finish the studio early, you might want to sharpen your SQL skills. You never know when you will be asked to compose a hard SQL query. Here are some queries you might want to try against the movie database. The documentation and the SQL table descriptions of the movie database can be found in the course web site.

Query 1: Find the stage name of each actor who has been in a movie that has been remade at least once.

RESULT: 2907 tuples

Query 2: Find all films with award winning casts, famous quotes and that have been remade.

RESULT:

Casablanca
Gone with the Wind
The Philadelphia Story

Query 3: Find all actors who have quotes in more than one film and at least one special award.

RESULT:

Marlene Dietrich

Query 4: Find all films that have actors who have won more than one special award in years after their second film was released.

RESULT:

Returns 68 tuples.

Query 5: Find everyone who has won a special award since 1990.

RESULT:

Myrna Loy
Stanley Kramer