ORACLE Reference

1 Introduction

The ORACLE RDBMS is a relational database management system. A command language called SQL*PLUS (SQL stands for “Structured Query Language”) is used for working with an ORACLE database. The SQL*PLUS program accepts SQL*PLUS commands from your keyboard, executes them through the ORACLE RDBMS, and formats the results as you specify.

2 Downloading & Setting up

You can install Oracle Database 11g Express Edition (Oracle Database XE), which is an entry-level, small-footprint database based on the Oracle Database 11g Release 2 code base. It’s free to download.\(^1\)

To complete your CS 452 project assignments, you will need to use either Oracle Database 11g Express Edition (or MySQL 5.6 or any relational database system that is comparable with Oracle Database XE). You are also required to install and rely on Oracle SQL Developer, which is a free integrated development environment that simplifies the development and management of an Oracle database, to (i) interact with Oracle Database XE and (ii) print out the results of each query specified in the CS 452 project assignments.

3 ORACLE Commands

The commands you use to work with your ORACLE assignments are SQL*PLUS commands. SQL*Plus recognizes two kinds of commands:

- **SQL Commands** to manipulate the database. SQL commands may be more than one line long, and must be ended with a semicolon (;). The semicolon is a signal that you want SQL*Plus to process the command.

- **SQL*Plus commands** to edit, store, and retrieve SQL commands. SQL*Plus commands must be entered on a single line, and need not be ended with a semicolon.

Whenever you want to quit from ORACLE, type EXIT.

3.1 SQL Commands

Listed as follows are some of the basic SQL commands for creating new tables, appending rows in tables, joining different tables, and querying tables in an ORACLE database:

3.1.1 Creating a Table

In the CREATE TABLE command, you name the table (e.g., student) and its columns (e.g., ssno, sname, etc.) You specify what kind of information each column may contain (NUMBER for numbers, CHAR for char values, etc.), and indicate the maximum width of the values stored in each column.

\(^1\)Instructions for downloading Oracle Database XE can be found in the Oracle Express Edition Installation document posted under the CS452 Tutorial website (http://students.cs.byu.edu/~cs452ng/tutorial.html).
The following command created a sample ‘student’ table:

```sql
CREATE TABLE student
    (ssno NUMBER(9),
     sname CHAR(20),
     address CHAR(25),
     gpa NUMBER(5));
```

### 3.1.2 Describing a Table

The **DESC** or **DESCRIBE** command displays a brief description of a table. The description contains each column’s position, storage size, data type, and column names, etc.

For example, to describe the table ‘student’, enter:

```sql
DESC student;
```

### 3.1.3 Listing all Existing Tables

The following command displays all the tables (and views) that you own:

```sql
SELECT Tname FROM tab;
```

where ‘Tname’ retrieves the name of a table, among others, and ‘tab’ is an entry in the data dictionary.

### 3.1.4 Inserting Rows into a Table

The **INSERT** command contains values for columns in the table. Enter the values, separated by commas, in the same order as the columns are defined. Enclose constant char values in apostrophes (’).

For example, to add a new student to the ‘student’ table, enter:

```sql
INSERT INTO student
VALUES (123456789, 'Mark Anderson',
        '155 Denison, Manhattan', 3.75);
```

### 3.1.5 Retrieving Information from a Table

To retrieve information from one or more tables, use the SQL command **SELECT**. **SELECT** tells SQL*Plus which tables to retrieve information from, and which columns and row to retrieve.

For example, to list the social_security_number, name, address, and gpa of each student in the ‘student’ relation, enter:

```sql
SELECT ssno, sname, address, gpa FROM student;
```

or

```sql
SELECT * FROM student;
```
'*' makes SELECT display all columns of the table(s) specified (in the FROM clause), in the order they were defined when the table(s) were created.

To eliminate duplicate rows before they are displayed, use the **DISTINCT** clause in the SELECT command.

For example, to list all distinct student names, enter:

```
SELECT DISTINCT sname FROM student;
```

To select rows from a table, include a **WHERE** clause in the SELECT command. A SELECT command with a WHERE clause retrieves only those rows that meet the search conditions.

Suppose you want to list information about the student ‘Mark Anderson’, enter:

```
SELECT *
FROM student
WHERE sname = 'Mark Anderson';
```

When you use a character value in a WHERE clause, you must enclose it in single quotes. You do not need to enclose number values in quotes.

### 3.1.6 Copying Rows between Tables

You can use an INSERT command with a subquery to select rows from one table and insert them into another table. The subquery replaces the VALUES clauses. Only these columns and rows selected by the subquery will be inserted.

For example, suppose you want to append all student records with grade point average 3.70 or above in the table ‘student’ to the table ‘outstanding’ (assume that the table ‘outstanding’ has already been created), enter:

```
INSERT INTO outstanding (ssno, sname, address, gpa)
SELECT *
FROM student
WHERE gpa > 3.70;
```

By the way, you may also create a table and copy rows into it in a single operation by using the CREATE TABLE command with the **AS** clause. If you had created ‘outstanding’ and copied the rows into it in one operation, the CREATE TABLE command would have looked like this:

```
CREATE TABLE outstanding (ssno, sname, address, gpa)
AS (SELECT *
    FROM student
    WHERE gpa ≥ 3.70);
```

The columns’ types and widths of the table ‘outstanding’ are not specified: they are determined by the types and widths of the columns returned by the subquery.
3.1.7 Deleting Rows from a Table

To delete one or more rows from a table, you must select the rows with an appropriate condition in a WHERE clause.

For example, to delete all students with grade point averages below 2.0, enter:

```
DELETE FROM student
WHERE gpa < 2.0;
```

To delete all the rows from a table, enter the DELETE command *without* a WHERE clause.

3.1.8 Updating Rows in a Table

You can update one or more rows by selecting rows with the UPDATE's WHERE clause. All rows that satisfy the clause's condition will be updated.

For example, to update Mark’s current address, enter:

```
UPDATE student
SET address = '10 Anderson, J.C.'
WHERE sname = 'Mark Anderson';
```

3.1.9 Joining Different Tables

To make a query in which rows of two or more tables are joined, you must specify the join columns that contain corresponding information in the joined tables. Specify the tables to be joined in the SELECT commands’ FROM clause, and specify the join columns in the WHERE clause:

```
SELECT columns
FROM table1, table2, ..., table_n
WHERE condition;
```

For example, to find the employee Joe’s location, enter:

```
SELECT ename, loc
FROM employee, department
WHERE ename = 'Joe' AND employee.deptno = department.deptno;
```

The ‘deptno’ column name is prefixed by the table name ‘employee’ or ‘department’. This is because both ‘employee’ and ‘department’ have a column named ‘deptno’. When two columns have the same name, use table name prefixed to show exactly which columns you mean. If a column name is unique in the tables specified in a query, you need not prefix it.

3.1.10 Abbreviating a Table Name

Although table name prefixes prevent ambiguity in a query, they can be tedious to enter. You can define temporary labels in the FROM clause and use them in the remainder of the query. Such temporary labels are sometimes called table aliases.

To use table labels to abbreviate the table names in the previous example, enter:
SELECT ename, loc
FROM employee E, department D
WHERE ename = 'Joe' AND
    E.deptno = D.deptno;

3.1.11 Matching a Value in a List

The IN operator lets you select rows that match one of the values in a list which is retrieved from one or more tables. The NOT IN operator is the complement of the IN operator.

For example, to list the employees located in 'Chicago' with the same position as 'Joe', enter:

SELECT ename, loc, position
FROM employee E, department D
WHERE loc = 'Chicago' AND
    E.deptno = D.deptno AND
    position IN
        (SELECT position
         FROM employee
         WHERE ename = 'Joe');

3.1.12 Set Operators

A query (including a subquery) may be composed of two or more queries with the operators UNION, INTERSECT, and MINUS (i.e., set difference):

- **UNION** returns all distinct rows returned by *either* of the queries it applies to.
- **INTERSECT** returns all rows returned by *both* of the queries it applies to.
- **MINUS** returns all rows returned by the *preceding* query, but *not* by the *following* query.

The operators UNION, INTERSECT, and MINUS are most useful for constructing subqueries that refer to different tables. These operators also put certain restrictions on the queries that they operate on, i.e., the queries must select the same number of columns, and corresponding columns must be of the same type.

3.2 SQL*Plus Commands

Listed as follows are some of the basic SQL*Plus commands that allow you to examine, change, or rerun a SQL command without re-entering it. These commands are helpful if you mistype a command, or want to modify a query you have entered.

3.2.1 Listing the Current SQL Command

When you enter a SQL command, SQL*Plus stores it in the SQL buffer. The command in the buffer is called the *current SQL command*, and it stays there until you enter another SQL command.

To list the current SQL command in the SQL buffer, enter **L** (or **LIST**)

The asterisk after the line number shown on your screen indicates the current line. If you entered a CHANGE command at this point, the change would affect the current line.
To list a particular line for editing, enter \textbf{L} \textless \text{line number}\textgreater

\subsection*{3.2.2 Editing the Current SQL Command}

Suppose you mistakenly enter a wrong word in a command. Instead of re-entering the entire command, you can correct the mistake by editing the command in the buffer:

First, use the \texttt{LIST} command followed by the line number to display the line that contains the error.

Second, use the \texttt{CHANGE} command, which has the following format, to correct the mistake:

\begin{verbatim}
C (or CHANGE) /old/new /
\end{verbatim}

The \texttt{CHANGE} command finds the first occurrence in the current line of the ‘old’ character sequence to be changed, and changes it to the ‘new’ sequence.

\subsection*{3.2.3 Repeating the Current Command}

After you have changed a command, you can use the \texttt{R} or \texttt{RUN} command to run it again.

\subsection*{3.2.4 Adding a New Line}

To add a new line to the buffer, or to insert a line between two existing lines, use the \texttt{I} or \texttt{INPUT} command.

Suppose you want to add a new line next to line ‘n’. First, position at line ‘n’ and enter \texttt{INPUT} and the new line. Then press [Return] twice: once to indicate the end of the new line, and again to indicate that no more lines are being entered.

\subsection*{3.2.5 Deleting a Line}

To delete a line in the buffer, use the \texttt{DEL} command:

1. Use the \texttt{LIST} command to display the line you want to delete.
2. Enter \texttt{DEL}.

\subsection*{3.2.6 Saving a Command}

To save a command which is stored in the current buffer for later use, use the \texttt{SAVE} command. Follow the command with a file name:

\begin{verbatim}
SAVE file
\end{verbatim}

The suffix .SQL is added to the file name to identify it as a SQL query file unless there is already a period in the name you specify.

\subsection*{3.2.7 Retrieving a Command}

If you want to work with a stored command, you must retrieve the command from the file in which it is stored. Just as you can save a query from the buffer to a file with the \texttt{SAVE} command, you can retrieve a query from a file to the buffer with the \texttt{GET} command:

\begin{verbatim}
GET file
\end{verbatim}

The suffix .SQL is added to the file name unless there is already a period in the name you specify.
3.2.8 Executing a Command from a File
The **START** command retrieves a query from a file and runs it in a single step. Follow the word START with the name of the file:

```
START file
```
If there is no suffix to the file name, it is treated as a SQL query file which is suffixed by .SQL.

3.2.9 Editing a Command in a File
You can run the host computer’s text editor without leaving SQL*Plus by entering the **EDIT** command.

```
EDIT
```
EDIT edits the contents of the current buffer. When you tell the text editor to save edited text, the text is saved back into the current buffer.

You can also edit the contents of a file by entering EDIT followed by the name of the file:

```
EDIT file
```
Once again, the file name suffix .SQL is added to the name unless the name contains a period.

3.2.10 Clearing the Current Buffer
To delete all lines in the current buffer, enter:

```
CL BUFF
```