

## Lab 1 - Table Definition, Manipulation and Simple Queries

### 1. INTRODUCTION

Oracle is a relational database management system (DBMS) that stores, manages, and manipulates data. It consists of the database manager, the Database Administrator utility and other user utilities. The query language which we will use is called SQL (an acronym for Structured Query Language), and is implemented as part of a package called SQL\*Plus.

### 2. USING SQL\*PLUS

#### 2.1 Logging into the gateway server

Launch a SSH Secure Shell Client software. Click the “Quick Connect” button. Enter *gateway.cs.cityu.edu.hk* for Host Name and your student ID for User Name. Then click the Connect button. Answer Yes to the Host Identification question. Then enter your student ID as your login password. You can then see the server shell command prompt.

#### 2.2 Starting SQL\*Plus

To start SQL\*Plus, first login gateway server account and then type the following command:

***sqlplus***

The username will be your EID and the password will be your student id (not your CityU password).

SQL\*Plus will display the version of the Oracle Server:

***SQL\*Plus: Release 12.2.0.1.0 Production on Sat Feb 20 12:06:30 2021***

***SQL\*Plus: Release 12.2.0.1.0 Production on Sat Feb 20 12:06:30 2021  
Copyright (c) 1982, 2016, Oracle. All rights reserved.***

***Enter user-name: your\_EID***

***Enter password: your\_student\_ID***

***Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 -  
64bit Production With the Partitioning, OLAP, Data Mining and Real  
Application Testing options***

Then the SQL\* Plus command prompt ***SQL>*** will appear, which indicates that SQL\*Plus is ready to accept your commands.

#### 2.3 Exiting from SQL\*Plus

When you have finished working with SQL\*Plus and want to exit from the SQL\*Plus environment, enter the following command at the SQL\*Plus command prompt:

***SQL> exit OR SQL> quit***

You will be disconnected from Oracle and return to the UNIX prompt.

**Always remember to type *exit* or *quit* to terminate the SQL\*Plus session normally. Otherwise your modification to the database may not take effect and your data may be lost.**

## 2.4 Getting Help

You may get information on the commands and conventions of SQL\*Plus, SQL, and PL/SQL using **help** command:

***SQL > help***

The following command displays a list of SQL\*Plus, SQL and PL/SQL commands:

***SQL > help index***

Get help on the SQL\*Plus clause LIST:

***SQL > help list***

## 3. LAB OBJECTIVES

In this lab section, you will learn to:

Define tables

- Create your own tables
- Drop your own tables

Manipulate tables

- Insert tuples into your tables
- Altering your own tables

Query information from tables

- Select all the columns from a table
- Select specific columns from a table
- Creating a new table from existing data
- Updating data in tables
- Deleting tuples rows in tables

## 4. BASIC CONCEPT OF RELATIONAL DATABASES

A database is an organized collection of information. In a relational database, the information is organized in the form of tables. In this lab, we will cover two of the most important components of a database: tables and queries. Each student needs to create the following tables and insert records into the corresponding tables. Information stored in these tables is needed for all the laboratory exercise in this course.

The following set of tables contains the personal records for a fictitious company.

EMP                    contains information about the employees of the company  
DEPT                  contains information about the departments of the company  
SALGRADE          group salary ranges into grades

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

## SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

## EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7655	JONES	MANAGER	7839	2-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	1-MAY-91	2850		30
7782	CLARK	MANAGER	7839	9-JUN-81	2450		10
7788	SCOTT	ANALYST	7655	21-MAR-87	3000		20
7839	KING	PRESIDENT		12-NOV-81	5000	0	10
7844	TURNER	SALESMAN	7698	18-SEP-81	1500		30
7876	ADAMS	CLERK	7788	24-APR-87	1100		20
7900	JAMES	CLERK	7698	3-DEC-81	950		30
7902	FORD	ANALYST	7655	3-DEC-81	3000		20
7934	MILLER	CLERK	7782	3-JAN-81	1300		10

## 5. SIMPLE DEFINITION OF TABLES

### 5.1 Create Tables

To create a user-defined table, the CREATE TABLE command is available which defines the table's name, its columns/attributes, and other properties.

#### 5.1.1 Common Data Types

CHAR (n)

n = maximum length of char string

NUMBER (n,d)

n = maximum number of digits

d = maximum number of digits right of decimal.

DATE

The relevant data plus the actual time of day

LONG

Up to 65,536 characters (64K) may be stored per field

RAW

Raw binary data

### 5.1.2 Forbidding NULL Values

To forbid NULL values in a column, enter the NOT NULL clause at the end of the column information. For example, the command:

```
SQL > CREATE TABLE DEPT  
2 (DEPTNO NUMBER (2) NOT NULL,  
3 DNAME VARCHAR (15),  
4 LOC VARCHAR (15));
```

adds the table definition of DEPT to your database.

The SALGRADE and EMP tables are created as follows:

```
SQL > CREATE TABLE SALGRADE  
2 (GRADE NUMBER,  
3 LOSAL NUMBER,  
4 HISAL NUMBER);
```

```
SQL > CREATE TABLE EMP  
2 (EMPNO NUMBER (4) NOT NULL,  
3 ENAME VARCHAR (15),  
4 JOB VARCHAR (15),  
5 MGR NUMBER (4),  
6 HIREDATE DATE,  
7 SAL NUMBER (7,2),  
8 COMM NUMBER (7,2),  
9 DEPTNO NUMBER (2));
```

## 5.2 The DESCRIBE Command

You can use the DESCRIBE command to obtain a description of a specific table. The description will contain

Name -	name of column
Null? -	whether null values are allowed in this column
Type -	data type of column

```
SQL > DESCRIBE DEPT
```

## 5.3 Drop Tables

To destroy the table and its contents, use the DROP TABLE command:

```
SQL > DROP TABLE DEPT;
```

For the time being, don't use this command as you'll need the tables later on.

## 6. SIMPLE MANIPULATIONS

### 6.1 Inserting Tuples into Tables

After the table is created, you can use the INSERT INTO ... VALUES command to insert tuples/rows into your table. For example, the command:

***SQL > INSERT INTO DEPT***

***2 VALUES (10, 'ACCOUNTING', 'NEW YORK');***

adds the first tuple to the DEPT table.

You can use variables to facilitate the data insertion. For example, to insert rows into the DEPT table, you can type the following command:

***SQL > INSERT INTO DEPT***

***2 VALUES (&DEPTNO, '&DNAME', '&LOC');***

and just repeatedly invoke this command by typing / or the **RUN** command, and enter new values accordingly. For those entries that are “empty”, e.g. some COMM fields, type NULL as the input.

Note that your database may not be modified immediately after an insert, update, or delete operations. If you would like to ensure that these changes take effect immediately, you can run the “COMMIT” command:

***SQL> COMMIT;***

***Commit complete.***

## 6.2 Altering Tables

You may add a new column or change the length of an existing column by using the ALTER TABLE command with either the ADD or MODIFY parameters. Note that you can neither reduce the width of a column if the table already contains data, nor change the data type (unless the column is empty).

Change the length of column LOC in table DEPT from 15 to 20

***SQL > ALTER TABLE DEPT***

***2 MODIFY (LOC VARCHAR(20));***

## 6.3 Executing .sql Script Files

When creating a large database, it is often convenient to write all necessary commands (CREATE TABLE, INSERT, etc.) in a script file using your text editor of choice. Then, you can load the script file in sqlplus and all commands will be executed at once. You can do this by typing @ and then the file name.

For example, if you would like to execute all commands in the file called script.sql, you need to type the following command at the SQL prompt:

***SQL > @script.sql***

Note that here we assume that script.sql is located in the current directory where we started sqlplus.

To simplify creating the tables for this lab, we have added a script file “test.sql” to the tutorial section on Canvas. If you want to use the file, you will first need to copy this it to the gateway server using your SSH client.

## 7. SIMPLE QUERIES

To retrieve information from the database we use the SELECT command. The following starts with some simple usages of the SELECT command. We will look at more sophisticated ones in subsequent lab classes.

### 7.1 The SELECT Clause

A SELECT command must have at least two clauses:

**SELECT**      the columns to retrieve  
**FROM**        the tables which contain those columns

You may place the clauses on the same line or on separate lines. Always follow the last clause in the command with a semicolon (;). For example, the command:

***SQL > SELECT DNAME FROM DEPT;***

Displays the information in the department name column (DNAME) from the table DEPT. It could also be entered as:

***SQL > SELECT DNAME  
2 FROM DEPT;***

or even:

***SQL > SELECT DNAME  
2 FROM DEPT  
3 ;***

For the time being, we will only SELECT columns from a single table. We will look at joining tables in subsequent labs.

#### 7.1.1 Selecting All the Columns from a Table

(1) Select every column of the DEPT list the names of the columns:

***SQL > SELECT DEPTNO, DNAME, LOC  
2 FROM DEPT;***

If all the columns of a table are to be selected you may use an asterisk (\*) in place of the list of column names.

(2) Select all the columns from a table

***SQL > SELECT \* FROM DEPT;***

#### 7.1.2 Selecting Specific Columns from a Table

If only some of the columns are to be selected then list only the names of the columns that you want to see (i.e. recall from the lecture that these are called the “projection attributes”).

- Select just the DNAME and DEPTNO columns from the DEPT table.

***SQL > SELECT DNAME, DEPTNO FROM DEPT;***

The order in which the columns are listed in the SELECT clause controls the left-to-right sequence in which the columns are displayed. When an asterisk is used, the left-to-right sequence in which the columns are displayed is the order in which they were defined when the table was created.

- Select all columns from the EMP table.

***SQL > SELECT \* FROM EMP;***

Look at the results of the queries in this section. Notice that the result of any query is itself a table called the *result table* – made up of columns and rows.

- List all the jobs in the EMP table without eliminating duplicates.

**SQL > SELECT JOB FROM EMP;**

You can eliminate duplicate rows in the result by adding the DISTINCT keyword.

- List each job once that is in the EMP table.

**SQL > SELECT DISTINCT JOB FROM EMP;**

## 7.2 The WHERE Clause

In this subsection, you will learn about:

- Selecting specific rows from a table
- Selecting rows that satisfy multiple conditions
- Selecting rows that satisfy one of several conditions
- Selecting rows that DO NOT meet certain conditions
- Selecting rows using both ANDs and ORs – Boolean Precedence

### 7.2.1 Selecting Specific Rows from a Table

In the previous examples, all the rows of the table were selected. But suppose you want to retrieve only the employees in department 30. To do this, you will need to use a WHERE clause in your SELECT command. A WHERE clause, when used, always follows the FROM clause.

**SELECT**      some columns  
**FROM**        some tables  
**WHERE**        certain conditions are met

Try the following example:

- Select only the employees in department 30.

**SQL > SELECT \***  
**2 FROM EMP**  
**3 WHERE DEPTNO = 30;**

A WHERE clause causes a “search” to be made, and only those rows that meet the search-conditions are retrieved. In the example above, only those rows WHERE the employee’s department number is equal to 30 will be selected. That is, the WHERE clause was used to compare values stored in the DEPTNO column of the EMP table with the value 30 that you specified as a part of the query.

WHERE    DEPTNO            =                            30  
WHERE    Column-name    Comparison-operator    Constant-value

A WHERE clause can contain a numeric constant-value (30 in the example above) or a character constant-value shown as in the example below.

- List the names, numbers, and departments of all clerks

**SQL > SELECT ENAME, EMPNO, DEPTNO**  
**2 FROM EMP**  
**3 WHERE JOB = ‘CLERK’;**

If the constant value that you specify is a character data literal then it has to be enclosed in single quotes (e.g. ‘CLERK’). The case also matters: try the previous query, but replace ‘CLERK’ by ‘clerk’. Numeric data does not need to be enclosed in quotes.

A WHERE clause condition can use any of the following comparison operators:

=	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

- Find the names and department numbers of all departments with a department number greater than 20.

```
SQL > SELECT DNAME,DEPTNO  
2 FROM DEPT  
3 WHERE DEPTNO > 20;
```

- Find the names and department numbers of all departments with a department number greater than or equal to 20.

```
SQL > SELECT DNAME, DEPTNO  
2 FROM DEPT  
3 WHERE DEPTNO >=20;
```

In addition to a comparison with a constant-value, a WHERE clause can compare a value stored in one column of a row with a value in another column of the same row.

WHERE	SAL	>	COMM
WHERE	Column-name	Comparison-operator	Column-name

- Find the employees whose commission is greater than his salary.

```
SQL > SELECT ENAME, SAL, COMM  
2 FROM EMP  
3 WHERE COMM > SAL;
```

### 7.2.2 Selecting Rows That Satisfy Multiple Conditions

Sometimes it is necessary to specify more than one search-condition in a WHERE clause to retrieve the desired rows from a table.

- List the name, job, title and salary of all employees in department 20 that make more than \$2,000:

```
SQL > SELECT ENAME, JOB, SAL  
2 FROM EMP  
3 WHERE DEPTNO = 20  
4 AND SAL > 2000;
```

Here, the two search conditions are connected by the word AND. The AND means that both search-conditions must be true for a given row to be retrieved. Any number of search-conditions may be ANDed to a WHERE clause.

- Find all the salesmen in department 30 who have a salary greater than or equal to \$1,500.



```
SQL > SELECT ENAME, SAL, DEPTNO  
2 FROM EMP  
3 WHERE JOB = 'SALESMAN'  
4 AND DEPTNO = 30  
5 AND SAL >= 1500;
```

### 7.2.3 Selecting Rows That Satisfy One of Several Conditions

In addition to selecting rows that meet several conditions, you are also to select rows that meet one of several conditions.

- Find all the employees whose job is either manager or president.

```
SQL > SELECT ENAME, JOB, SAL  
2 FROM EMP  
3 WHERE JOB = 'MANAGER'  
4 OR JOB = 'PRESIDENT';
```

Here the search-conditions are connected by the word OR. OR means that if at least one of the two search-conditions is satisfied, then the row will be selected.

### 7.2.4 Selecting Rows that DO NOT Meet Certain Conditions

You can select rows that do not meet certain conditions as well as rows that do meet certain conditions.

- Find all managers who are NOT in department 30.

```
SQL > SELECT ENAME, SAL, COMM, JOB, DEPTNO  
2 FROM EMP  
3 WHERE JOB = 'MANAGER'  
4 AND DEPTNO != 30;
```

### 7.2.5 Selecting Rows Using Both AND's and OR's

AND and OR may be combined in the same query to specify as many search conditions as are necessary to select the desired rows.

- Find everyone whose job title is manager, and all the clerks in department 10.

```
SQL > SELECT *  
2 FROM EMP  
3 WHERE JOB = 'MANAGER'  
4 OR JOB = 'CLERK' AND DEPTNO = 10;
```

The WHERE clause in the above query can be written using parentheses without changing the meaning of query.

- Find all the manager (in any department), and all the clerks in department 10.

```
SQL > SELECT *  
2 FROM EMP  
3 WHERE JOB = 'MANAGER'  
4 OR (JOB = 'CLERK' AND DEPTNO = 10);
```

Parentheses are used to group search-conditions together to control the sequence in which the search-conditions are applied. This is called logical precedence. Depending on position of the parentheses, the meaning of the WHERE clause may be changed.

- Issue the same query again with the parentheses grouping the first two search conditions inserted of the last two.

```
SQL > SELECT *  
2 FROM EMP  
3 WHERE (JOB = 'MANAGER' OR JOB = 'CLERK')  
4 AND DEPTNO = 10;
```

Notice how in the last query both managers and clerks have to be in department 10, not just the clerks. When a WHERE clause contains more than two search conditions you should use parentheses to “classify the meaning” of the WHERE clause.

Any group of search-conditions can be *negated* by enclosing them in parentheses and preceding them with a NOT.

- Find everyone who is neither a manger nor a clerk, but is in department 10.

```
SQL > SELECT *  
2 FROM EMP  
3 WHERE NOT (JOB = 'MANAGER' OR JOB = 'CLERK')  
4 AND DEPTNO = 10;
```

## **8. DATA MANIPULATION**

### **8.1 Creating a New Table From Existing Data**

You can create a new table containing data from existing tables.

- Create a new table MY\_EMP from existing table EMP.

```
SQL > CREATE TABLE MY_EMP AS  
2 SELECT * FROM EMP;
```

### **8.2 Updating Data in Tables**

The update statement changes all or part of an existing row in a single table.

- Update the salary of the President in the MY\_EMP table.

```
SQL > UPDATE MY_EMP  
2 SET SAL = SAL *1.1  
3 WHERE JOB = 'PRESIDENT';
```

### **8.3 Deleting Tuples in Tables**

The delete statement remove rows from a single table.

- Delete the tuples in MY\_EMP where the employee’s JOB is CLERK.

```
SQL > DELETE FROM MY_EMP  
2 WHERE JOB = 'CLERK';
```