CSE 4308
DBMS LAB
Documentations

Prepared by,
Sabbir Ahmed
Lecturer,
Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)
Board Bazar, Gazipur-1704
Bangladesh.
Email: iamsabbirahmed12345@gmail.com, sabbirahmed@iut-dhaka.edu
Contact: +8801754221481, +8801630210656

Contents

1. Getting started with Oracle 10g XE

1.1. Connecting to the Database

Throughout the course we will use Oracle 10g XE database.
During the installation a password will be required. Be careful to remember that!

- How to open SQL Command Prompt?

  Just write *'sqlplus'* in the command prompt!
  Or,
  Search '*run SQL command line'* in the program search bar.

- How to connect to a database?

  Open SQL, write
  CONNECT USERNAME/PASSWORD;
  Or
  CONNECT

  [Then there will be prompt for username. Input the *username*. Then it will ask for *password*. When you will type the password, the characters will not be shown as oracle keeps it hidden. After finishing the password hit enter and see the system will be connected.]

- The default username for login is *'system'*, and the password is the one that you will give while installation. (We will learn later how to create a user manually.)

- For example, if the password is 'test123' then the command will be

  CONNECT SYSTEM/test123

- Along with 'system', Oracle creates many other users by default which are stored in a table named 'dba_users'. How can we see what's inside that table?

  To access any information inside the database, you must be connected to the database.
  To see the columns with datatype of the table 'dba_users' we have to write
  DESC DBA_USERS;

```
SQL> DESC DBA_USERS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USERNAME                                  NOT NULL VARCHAR2(30)
 USER_ID                                   NOT NULL NUMBER
 PASSWORD                                           VARCHAR2(30)
 ACCOUNT_STATUS                            NOT NULL VARCHAR2(32)
 LOCK_DATE                                          DATE
 EXPIRY_DATE                                        DATE
 DEFAULT_TABLESPACE                        NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE                      NOT NULL VARCHAR2(30)
 CREATED                                   NOT NULL DATE
 PROFILE                                   NOT NULL VARCHAR2(30)
 INITIAL_RSRC_CONSUMER_GROUP                        VARCHAR2(30)
 EXTERNAL_NAME                                      VARCHAR2(4000)
```

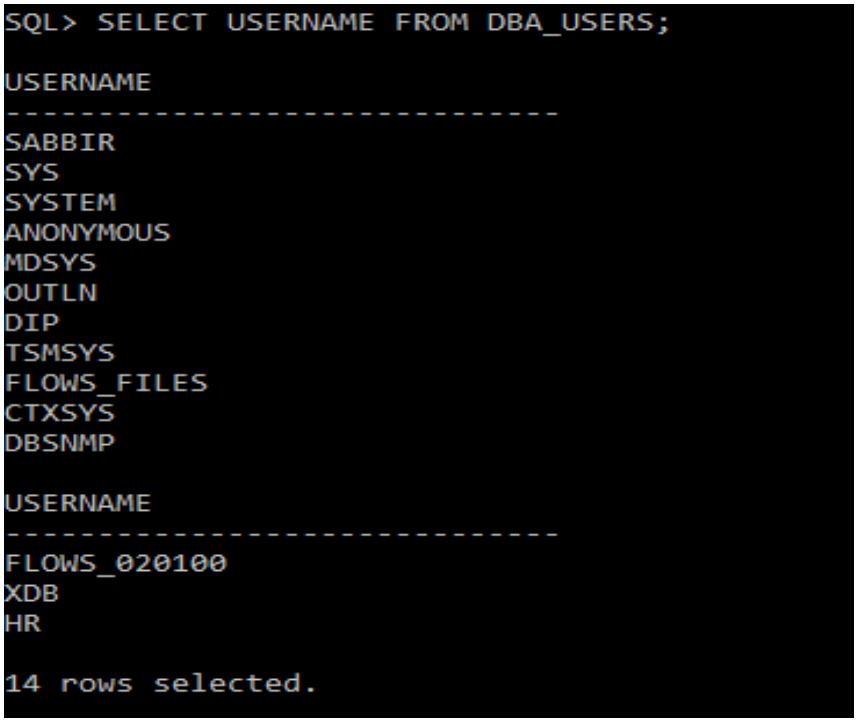
Here the first column ('name') contains the name of the columns of the table. 'Null?' signifies whether that particular column can have NULL value of not (NULL value means no value for that particular column). Finally, 'type' tells us the type of value that can be stored by the particular column.
A lot of information!

Still we haven't looked at what's inside the table. '*DESC DBA_USERS*' command gives us description about the table.
Here '*DESC*' tell Oracle to display the information about the name of the table mentioned after that. The generic command structure is *'DESC table_name'*.

[Note: In Oracle, all commands are case-insensitive. That means 'DESC DBA_USERS' and '*desc dba_users'* means the same. Throughout the handout, the commands are written in uppercase for better readability.]

Let's have a look inside the table.
Suppose you want to see the existing usernames. For this we need to write a Query using Structured Query Language (SQL). We will talk a lot about SQL in the later chapters.

```
SELECT USERNAME FROM DBA_USERS;
```



Here we can see a lot of users created by default (The top user is created by me.)

SQL allows us to do a lot of operations on data. For example, looking at the image we can see that the usernames are not sorted alphabetically. To do that we have to write:

```
SELECT USERNAME FROM DBA_USERS ORDER BY USERNAME;
```

```
SQL> SELECT USERNAME FROM DBA_USERS ORDER BY USERNAME;

USERNAME
------------------------------
ANONYMOUS
CTXSYS
DBSNMP
DIP
FLOWS_020100
FLOWS_FILES
HR
MDSYS
OUTLN
SABBIR
SYS

USERNAME
------------------------------
SYSTEM
TSMSYS
XDB

14 rows selected.
```

See how the usernames are sorted!
The query mentioned above can also be written as:

```
SELECT USERNAME
FROM DBA_USERS
ORDER BY USERNAME;
```

Actually, until the semicolon (;) is given, everything is considered as a single line. We'll follow this later style for better readability.

Moving back to the table description, we can see that there is a column named 'CREATED' which tells us the date on which the users are created.

To find out which user was created in which date, the Query is:

```
SELECT USERNAME, CREATED
FROM DBA_USERS
ORDER BY CREATED;
```

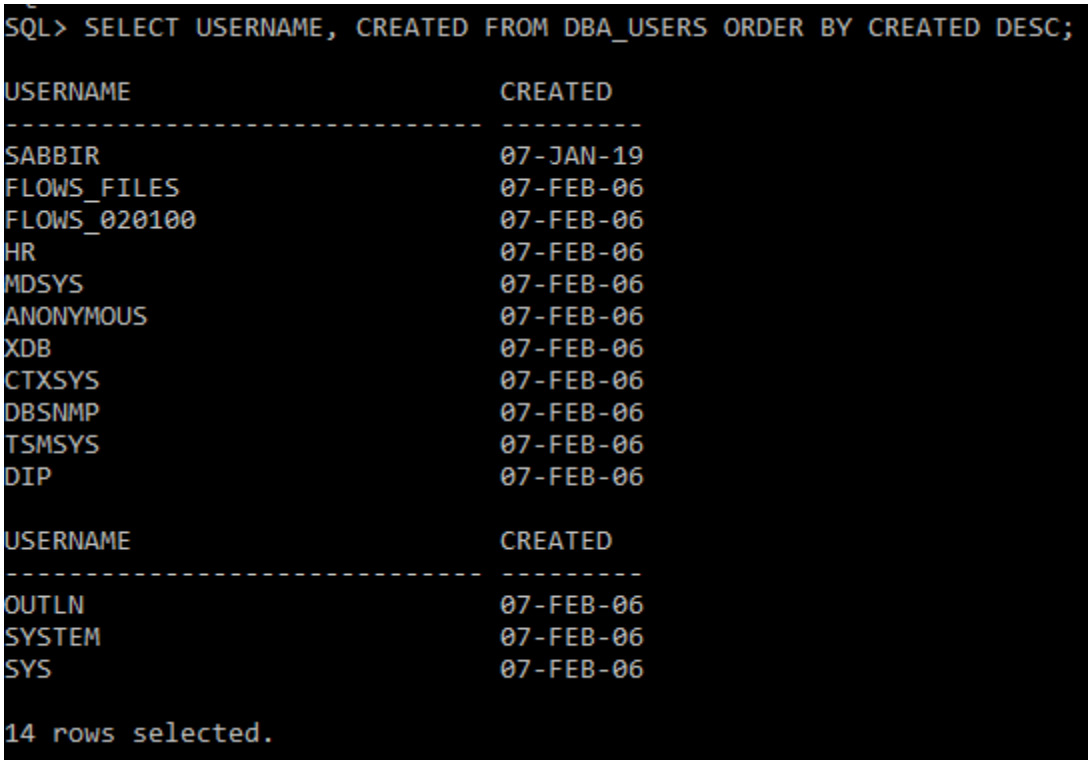
[Note: This will explicitly sort the result according to CREATED column. Everything is sorted in Ascending order by default.]

```
SQL> SELECT USERNAME, CREATED FROM DBA_USERS ORDER BY CREATED;

USERNAME                       CREATED
------------------------------ ---------
SYS                            07-FEB-06
SYSTEM                         07-FEB-06
OUTLN                          07-FEB-06
DIP                            07-FEB-06
TSMSYS                         07-FEB-06
DBSNMP                         07-FEB-06
CTXSYS                         07-FEB-06
XDB                            07-FEB-06
ANONYMOUS                      07-FEB-06
MDSYS                          07-FEB-06
HR                             07-FEB-06

USERNAME                       CREATED
------------------------------ ---------
FLOWS_020100                   07-FEB-06
FLOWS_FILES                    07-FEB-06
SABBIR                         07-JAN-19

14 rows selected.
```

The default order is ascending. To make it appear in descending order you have to use the following query:

```
SELECT USERNAME, CREATED
FROM DBA_USERS
ORDER BY CREATED DESC;
```

(Here DESC stands for Descending).



```
SQL> SELECT USERNAME, CREATED FROM DBA_USERS ORDER BY CREATED DESC;

USERNAME                       CREATED
------------------------------ ---------
SABBIR                         07-JAN-19
FLOWS_FILES                    07-FEB-06
FLOWS_020100                   07-FEB-06
HR                             07-FEB-06
MDSYS                          07-FEB-06
ANONYMOUS                      07-FEB-06
XDB                            07-FEB-06
CTXSYS                         07-FEB-06
DBSNMP                         07-FEB-06
TSMSYS                         07-FEB-06
DIP                            07-FEB-06

USERNAME                       CREATED
------------------------------ ---------
OUTLN                          07-FEB-06
SYSTEM                         07-FEB-06
SYS                            07-FEB-06

14 rows selected.
```

1.2. Creating a new user

- How to create a user in Oracle?

  CREATE USER *username* IDENTIFIED BY *password*

  For example, if we want to create a user named 'IUT' with password 'iutcse' the command should be:

  CREATE USER IUT IDENTIFIED BY iutcse

```
SQL> CREATE USER IUT IDENTIFIED BY iutcse;

User created.
```

- Now let's get connected to the database using this new user:

  CONNECT IUT/iutcse

```
SQL> conn iut/iutcse
ERROR:
ORA-01045: user IUT lacks CREATE SESSION privilege; logon denied


Warning: You are no longer connected to ORACLE.
SQL> CONNECT IUT/iutcse
ERROR:
ORA-01045: user IUT lacks CREATE SESSION privilege; logon denied
```

Why did this happen!

Each new user in oracle needs permission/privileges to do certain operations. Our new user *IUT* doesn't have any permission granted by any super-user (e.g.: *SYSTEM/SYS*) to log into the system. So the ERROR message has occurred.

Some examples of privileges are:

CREATE SESSION
CONNECT
RESOURCE (enable the user to create table/ manipulate data)
DBA (database administration privilege) and so on.

- How to enable the user to connect to the system?

```
SQL> GRANT CREATE SESSION TO IUT;

Grant succeeded.

SQL> CONNECT IUT/iutcse
Connected.
```

Some similar commands can be:

GRANT CREATE SESSION TO *username* WITH ADMIN OPTION;

To give both create session and resource privilege:

GRANT CREATE SESSION, RESOURCE TO *username;*

To give all possible privilege at a single command:

GRANT ALL PRIVILEGES TO *username*;

[Note: This is strictly NOT RECOMMENDED as the new user might get some unwanted privileges!]

- How to change password of a user?

ALTER USER *username* IDENTIFIED BY *password*;

But for that you have to be logged in as SUPERUSER! (SYSTEM/SYS/new user who has such privilege!)

- How to alter the system password in case of you have forgotten it?

CONN / AS SYSDBA
ALTER USER *username* IDENTIFIED BY *new_password*;
Or

CONN SYSTEM / *password* AS SYSDBA
ALTER USER *username* IDENTIFIED BY *new_password*;

- To Drop (delete) a user:

DROP USER *username* CASCADE;

Here cascade means all the objects (for example: tables) will also be deleted with the removal of the user.

- Let’s have a look at the power of SQL once again:

  We can format time in different ways using SQL. Some examples are showed below:

```
SQL> select to_char(created, 'DD-MON-YYYY HH24:MI:SS') from dba_users;

TO_CHAR(CREATED,'DD-
--------------------
23-JAN-2018 00:52:15
23-JAN-2018 00:42:39
07-FEB-2006 22:10:13
07-FEB-2006 22:10:13
07-FEB-2006 22:40:15
07-FEB-2006 22:44:47
07-FEB-2006 22:10:24
07-FEB-2006 22:17:03
07-FEB-2006 22:27:15
07-FEB-2006 22:52:43
07-FEB-2006 22:38:38

TO_CHAR(CREATED,'DD-
--------------------
07-FEB-2006 22:35:21
07-FEB-2006 22:52:43
07-FEB-2006 22:40:14
07-FEB-2006 22:51:21

15 rows selected.
```

2. Manipulating Tables

2.1. Creating Table

- How to create a table?

```
CREATE TABLE DEPT (
        DEPTID NUMBER,
        DEPTNAME VARCHAR2 (20)
);
```

[Note: Remember that SQL isn't case sensitive! But the values are.]

Here some problems might occur if the user is not given sufficient privilege!

Remember our user IUT. SYSTEM gave CREATE SESSION privilege to the user. So now s/he can log in. But while trying to create table:

```
SQL> CREATE TABLE DEPT (
  2  DEPTID NUMBER,
  3  DEPTNAME VARCHAR2 (20)
  4  );
CREATE TABLE DEPT (
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

How to solve this issue?

SYSTEM must give the user *IUT* permission to create table. This permission comes with *RESOURCE* privilege.

```
SQL> CONN SYSTEM/iutcse
Connected.
SQL> GRANT RESOURCE TO IUT;

Grant succeeded.
```

[Note: Make sure that you are connected as a SUPERUSER at the time of giving privilege.]

Now as the user *IUT* has permission to create table, the statements will work.

```
SQL> conn iut/iutcse
Connected.
SQL> CREATE TABLE DEPT (
  2  DEPTID NUMBER,
  3  DEPTNAME VARCHAR2 (20)
  4  );

Table created.
```

- To see the column names and their types in SQL:

  DESC TABLE_NAME;

```
SQL> DESC DEPT
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 DEPTID                                             NUMBER
 DEPTNAME                                           VARCHAR2(20)
```

- How to show all the tables that are under a particular user?

  DESC DBA_TABLES;
  [Note: DBA_TABLES is a table containing information about all the tables under each user. To access this table, user must be connected as *'system'* or any user with admin privilege.]

  SELECT TABLE_NAME
  FROM DBA_TABLES
  WHERE OWNER= *'username'*;

```
SQL> SELECT TABLE_NAME
  2  FROM DBA_TABLES
  3  WHERE OWNER='IUT';

TABLE_NAME
------------------------------
DEPT
DEPT1
```

Here we can see, there are two tables under the user named '*iut*'.

- Suppose the user wants to create a new table named '*DEPT*' but not sure whether the table is already created or not. If it is already existing, oracle will not let the user create a new table with the same name.

  How to be sure that whether a table called '*DEPT*' is existing or not?

  One solution might be deleting the existing table.

  ```
  DROP TABLE TABLE_NAME;
  ```

  But this is bad practice as the information might be necessary as the information of one table might be related to another table in a large system. So we can search using the following query:

  ```
  SELECT TABLE_NAME
  FROM DBA_TABLES;
  ```

  But it will return a long list of the existing tables. It is difficult to search my desired table_name from the list.

  Another way is:

  ```
  SELECT TABLE_NAME
  FROM DBA_TABLES
  WHERE TABLE_NAME='DEPT';
  ```

  IF the DEPT table is existing, some rows will be selected from the table.

  [Note: For accessing the '*DBA_TABLES*' table, the user must have admin privilege.]

2.2. Inserting data

- How to insert values to a table?

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

For example, lets insert some values in the DEPT table

```
INSERT INTO DEPT VALUES(1,'CSE');
INSERT INTO DEPT VALUES(2,'MCE');
INSERT INTO DEPT VALUES(3,'EEE');
INSERT INTO DEPT VALUES(4,'CEE');
INSERT INTO DEPT VALUES(5,'TVE');
INSERT INTO DEPT VALUES(6,'BTM');
```

- You can also specify the column names:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

For example, the above values could also be inserted as

```
INSERT INTO DEPT (DEPTID, DEPTNAME) VALUES(1,'CSE');
```

- How to see all the values from a table?

```
SELECT *
FROM table_name;
```

For example, to select all the values from the DEPT table, the query should be:

```
SELECT *
FROM DEPT;
```

```
SQL> SELECT * FROM DEPT;

    DEPTID DEPTNAME
---------- --------------------
         1 CSE
         2 MCE
         3 EEE
         4 CEE
         5 TVE
         6 BTM

6 rows selected.
```

2.3. Adding Constrains

Observing the previously created '*dept*' table, we find that there might be many rows with the same *DEPTID* or the same *DEPT_NAME* which is undesired!

To avoid such situation, we need to add some rules to the table which are called *constraints*.

2.3.1. Adding Primary Key

Primary key is a special column which is able to uniquely identify each record.

For example, in each class the students have their own *student_id* by which a student can easily be identified. This can be selected as primary key.
On the other hand, *student_name* might be same for many students. So this column is a bad choice for a primary key.

Example:

```
CREATE TABLE STUDENT (
ID NUMBER,
NAME VARCHAR2(30),
DEPT_NAME VARCHAR2(5),
PRIMARY KEY(ID)
);
```

Syntax:

PRIMARY KEY (*column_name*);
[If we want a particular column as a primary key.]

PRIMARY KEY (*column1, column2, … column_n*);
[Note: For depending on more than one column for uniquely identifying the records. For example, suppose we have a table '*music*' for storing music information. The columns of the table are *track_number*, *album_name*, *track_name*, *duration*, *singer*. Now here neither of the column can uniquely identify the records alone. But if we consider *track_number* and *album_name* as a pair, there should be one entry in the table. So we will consider this two column as key here. This type of key is commonly known as '*COMPOSITE KEY*'.

PRIMARY KEY (*track_number, album_name);*]

CONSTRAINT *constraint_name* PRIMARY KEY (*column1, column2, … column_n*);
[Number of column can be from one to n.]

- Why do we need constraint name?

  Let's consider the above mentioned table *'student'*:

  INSERT INTO STUDENT VALUES (1, 'TAMIM', 'CSE');
  INSERT INTO STUDENT VALUES (2, 'KAYES', 'CSE');
  INSERT INTO STUDENT VALUES (1, 'MOMINUL', 'CSE');

```
SQL> INSERT INTO STUDENT VALUES(1, 'TAMIM', 'CSE');

1 row created.

SQL> INSERT INTO STUDENT VALUES(2, 'KAYES', 'CSE');

1 row created.

SQL> INSERT INTO STUDENT VALUES(1, 'MOMINUL', 'EEE');
INSERT INTO STUDENT VALUES(1, 'MOMINUL', 'EEE')
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.SYS_C004024) violated
```

  Have a look at the last line, the data is not inserted as there is already a student with ID=1. So the error tells us that the unique constraint is violated.

  Now let's consider a table with *constraint_name* for the primary key and see the difference in the error message:

```
SQL> CREATE TABLE STUDENT(
  2  ID NUMBER,
  3  NAME VARCHAR2(30),
  4  DEPT_NAME VARCHAR2(5),
  5  CONSTRAINT student_id_pk PRIMARY KEY(ID)
  6  );

Table created.

SQL> INSERT INTO STUDENT VALUES(1, 'TAMIM', 'CSE');

1 row created.

SQL> INSERT INTO STUDENT VALUES(2, 'KAYES', 'CSE');

1 row created.

SQL> INSERT INTO STUDENT VALUES(1, 'MOMINUL', 'EEE');
INSERT INTO STUDENT VALUES(1, 'MOMINUL', 'EEE')
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.STUDENT_ID_PK) violated
```

Here the error message contains the name of the constraint that is being violated. This is really helpful for working with large system with a lot of constraints. It is directly pointing to the particular constraint and making it easy to handle or debug.

[Note: Don't get confused looking at the constraint name. It can be any name that you prefer.]

- How to add primary key to a table which is already created, but suppose u have forgotten to add the primary key?

  Creating a table without primary key:

```
CREATE TABLE STD (
SID NUMBER,
SNAME VARCHAR (20),
);
```

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name PRIMARY KEY (column_name);
```

  For example, here the code will be:

```
ALTER TABLE STD
ADD CONSTRAINT STD_SID_PK PRIMARY KEY(SID);
```

2.4. Updating values:

Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Example:

```
UPDATE CUSTOMERS
SET CONTACTNAME = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CUSTOMERID = 1;
```

2.5. Deleting values

- Deleting rows satisfying certain conditions:

  Syntax:

  DELETE FROM *table_name*
  WHERE *condition*;

  Example:

  DELETE FROM Customers
  WHERE CUSTOMERNAME='Alfreds Futterkiste';

- Deleting an entire table:

  DROP TABLE *table_name;*

- Deleting a user:

  DROP USER *username;*
  [Note: The user must have this permission.]

3. FAQs:

3.1. How to find which user is connected at this moment?

SHOW USER;

```
SQL> conn iut/iutcse
Connected.
SQL> show user
USER is "IUT"
```

3.2. How to disconnect/ log out?

Just write
DISC
Or
DISCONNECT

```
SQL> disc
Disconnected from Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production
SQL> DISCONNECT
SQL> CONN SYSTEM/iutcse
Connected.
SQL> DISCONNECT
Disconnected from Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production
```

3.3. What to do if the account is locked?
[The ERROR will occur as the picture shows]

```
Enter user-name: scott
Enter password:
ERROR:
ORA-28000: the account is locked
```

Solution:
CONNECT SYS/SYS AS SYSDBA
ALTER USER *username* IDENTIFIED BY *correct_password* ACCOUNT UNLOCK;