SQL STATEMENTS NEEDED IN DBS201

Use the following EMPLOYEE and DEPT given on page 2:

SELECTING DATA

- Selecting all rows and columns:

```
SELECT        *
FROM  EMPLOYEE
```

- Selecting only certain rows but all columns:

```
SELECT    *
FROM EMPLOYEE
WHERE    EMPID > 300
AND   EMPID < 750
```

- Selecting certain columns:

```
SELECT    EMPID, LNAME
FROM EMPLOYEE
```

- Selecting using comparison operator:

```
SELECT EMPID
FROM EMPLOYEE
WHERE LNAME LIKE `A%`

SELECT LNAME
FROM EMPLOYEE
WHERE EMPID IN (10,20,30)

SELECT LNAME
FROM EMPLOYEE
WHERE EMPID BETWEEN 10 AND 20


SELECT EMPID
FROM EMPLOYEE
WHERE DEPTNO IS NOT NULL
```

Selecting data from more than one table **(JOIN)**

```
SELECT      E.EMPID,              can be done on one line or as shown here
            E.FNAME,              these are different styles but all READABLE
            E.LNAME,
            D.DNAME
FROM        EMPLOYEE   E,
            DEPT       D
WHERE     E.DEPTNO = D.DEPTNO   important condition on join
```

Change column names because it is going to users.

```
SELECT    EMPID AS "Employee No.",
          FNAME as "First Name"
FROM      EMPLOYEE
```

TO CREATE A TABLE

- With no constraints

What do you want to do? CREATE an object
What type of object? TABLE
What name? EMPLOYEE

What attributes/fields/columns does it have?

- Name the column and give the datatype and size followed by a comma
- Apply the constraints in one of several ways as shown in the other examples

```
CREATE    TABLE EMPLOYEE
          (EMPID          NUMBER (4),
          FNAME           CHAR (20),
          LNAME           CHAR (20),
          SALARY          DECIMAL (6,2),
          COMMISSION      DECIMAL (6,2),
          COMM-RATE DECIMAL (4,3),
          GENDER          CHAR,
          STARTDATE       DATE,
          DEPTNO          CHAR (2)
          )
```

Full length is 4, the scale is 2 decimal positions to the right EXAMPLE WOULD BE 0.025

Used for fixed character length data

The possible data types are
CHAR
INTEGER
NUMBER
DECIMAL
DATE there is no length specified for date
… And many others not covered on this course (BLOB, BINARY, CLOB ….)

CREATE TABLE WITH CONSTRAINTS

- With constraints in line with column definition

```
CREATE    TABLE EMPLOYEE
          (EMPID          NUMBER (4)          PRIMARY KEY,
          FNAME           CHAR (20)     NOT NULL,
          LNAME           CHAR (20)     NOT NULL,
          SALARY          DECIMAL (6,2),
          COMMISSION      DECIMAL (6,2),
          COMM-RATE DECIMAL (4,3)       CHECK (RATE IN (0.015, 0.020, 0.025)),
          GENDER          CHAR                CHECK (GENDER = 'M' OR 'F'),
          STARTDATE       DATE,
          DEPTNO          CHAR (2)
          )
```

- With some of the constraints shown, but defined out of line (at the bottom)

```
CREATE      TABLE EMPLOYEE
        (EMPID              NUMBER (4),
        FNAME               CHAR (20),
        LNAME               CHAR (20),
        SALARY              DECIMAL (6,2),
        COMMISSION          DECIMAL (6,2),
        COMM-RATE DECIMAL (4,3),
        GENDER              CHAR,
        STARTDATE           DATE,
        DEPTNO              CHAR (2),
        CONSTRAINTEMPLOYEE_EMPID_PK
                            PRIMARY KEY (EMPID),
        CONSTRAINTEMPLOYEE_COMM_RATE_CK
                            CHECK (RATE IN (0.015, 0.020, 0.025))
        )

        CREATE      TABLE DEPT
        (DEPTNO                     CHAR (2) PRIMARY KEY,
        DNAME                       CHAR (20) NOT NULL )
```

Create a table that will handle the following table definition

CUSTOMER (CID, LAST, FIRST, STREET, CITY, PROV, PCODE, BALANCE, CREDIT_LIMIT, SID)

Table name: CUSTOMER_____

Column	Data Type	SIZE	PK	FK and reference	NN	UK	CK and default
CID	NUMERIC	3	Y				
LAST	CHAR	20			Y		
FIRST	CHAR	20			Y		
STREET	CHAR	60			Y		
CITY	CHAR	20			Y		Toronto
PROV	CHAR	2			Y		ON
PCODE	CHAR	6			Y		
BALANCE	DECIMAL	7.2					
CREDIT_LIMIT	DECIMAL	7.0					See below
SID	NUMERIC	2		SALES_REP (SID)	Y		

```
CREATE TABLE    CUSTOMER
        (CID            NUMERIC (3) PRIMARY KEY,
        LAST            CHAR (20)   NOT NULL,
        FIRST           CHAR (20)   NOT NULL,
        STREET          CHAR (60)   NOT NULL,
        CITY            CHAR (20)   NOT NULL WITH DEFAULT 'Toronto',
        PROV            CHAR (2)            DEFAULT 'ON',
        PCODE           CHAR (6)            NOT NULL,
        BALANCE         DECIMAL (7,2),
        CREDIT_LIMIT DECIMAL (7,0)
                        CHECK (CREDIT_LIMIT IN (1000, 3000, 5000, 10000)),
        SID             NUMERIC (2),
```

```
    CONSTRAINT    CUSTOMER_SID_FK
                  FOREIGN KEY (SID)
                  REFERENCES SALES_REP (SID)
    );
```

CHANGING THE TABLE CHARACTERISTICS

If you add a column you need to add the name, datatype and size.

```
ALTER TABLE EMPLOYEE
    ADD COLUMN        BONUS DECIMAL (5,2);
```

If the column has a constraint it has to be done with another ALTER statement.

```
ALTER TABLE EMPLOYEE
    ADD CONSTRAINT EMPLOYEE_BONUS_CK CHECK (BONUS > 20.00);
```

How to handle a concatenated primary key
The same can be done in the create table starting with the word CONSTRAINT.

```
ALTER TABLE EMPLOYEE_SKILL
    ADD CONSTRAINT EMPLOYEE_SKILL_EMPID_SKILLID_PK
    PRIMARY KEY (EMPID, SKILLID);


ALTER TABLE DEPT
    ADD CONSTRAINT DEPT_LOCATION_UN
    UNIQUE (LOCATION);

ALTER TABLE EMPLOYEE
    ADD CONSTRAINT EMPLOYEE_DEPTNO_FK
    FOREIGN KEY (DEPTNO)
    REFERENCES DEPT (DEPTNO);
```

To remove a constraint

```
ALTER TABLE DEPT30
    DROP CONSTRAINT
    DEPT30_LOCATION_UK;
```

To remove a column

```
ALTER TABLE DEPT30
    DROP COLUMN CITY;
```

TO INSERT DATA

Suppose the table DEPT has the following columns

```
DEPTNO      CHAR (2),
DNAME       CHAR (20) NOT NULL,
LOCATION    CHAR (20),  I am allowing NULL although that would be unlikely in practice.
FLOOR       INTEGER
```

- To insert when you have when you only want to insert data into some of the fields. To do this the other fields or columns must allow NULL

```
INSERT INTO DEPT (DEPTNO, DNAME, FLOOR)
VALUES            ('AC', 'ACCOUNTING', 3);
```

Since location can accept NULL this could have been written as

```
INSERT INTO DEPT (DEPTNO, DNAME, LOCATION, FLOOR)
VALUES           ('AC', 'ACCOUNTING', NULL, 3);
```

Or it could have been written

```
INSERT INTO DEPT
VALUES           ('AC', 'ACCOUNTING', NULL,3);
```

- The3 column names do not have to be mentioned if you are loading data into all the columns and in the same order as they exist in the table.

Suppose 45,000 rows of data were coming in from another source but the original data is stored in a different order, then the INSERT would look like this.

```
INSERT INTO DEPT (LOCATION, DEPTNO, DNAME, FLOOR)
VALUES           ('BUILDING B', 'AC', 'ACCOUNTING', 3);
```

If the data is coming from another table you can use a select

```
INSERT INTO DEPT (DEPTNO, DNAME, LOCATION, FLOOR)
VALUES      as    SELECT DNO, NAME, BUILDING, FLOOR
                  FROM  OLD D_DEPT_TABLE;
```

DELETING DATA

Deleting ALL data from the table EMPLOEE

DELETE FROM EMPLOYEE: notice that no condition was specified

- Notice you don't have to say the word TABLE since you can't delete part of a collection or other object. Delete is only about tables.

Deleting some of the data

```
DELETE FROM DEPT
WHERE       DEPTNO = 'AC';
```
 will delete only rows with a department number of AC

```
DELETE FROM EMPLOYEE
WHERE STARTDATE < '01-JAN-2003';
```
 the format for date is site specific and is a character field

REMOVING THE TABLE

Removing the table also removes all the data. Deleting all the data (above) removes the data but leaves the structure of the table.

```
DROP TABLE EMPLOYEE;
```

Suppose the column DEPTNO in the table EMPLOYEE has a foreign key pointing to the table DEPT. If you were to drop the DEPT table then all the foreign keys in EMPLOYEE would have nothing to point to. You would

You first have to drop EMPLOYEE table or disable the foreign key constraint before you can drop the DEPT table.

The same applies if you just delete a row in the DEPT table and a foreign key references that row.

CREATING A VIEW

```
CREATE TABLE EMPLOYEE
 (EMPID  decimal(4)  PRIMARY KEY,
 FNAME  CHAR (20) NOT NULL,
 LNAME  CHAR (20) NOT NULL,
 SALARY  DECIMAL (6,2),
COMMISSION DECIMAL (6,2),
COMMRATE DECIMAL (4,3) CHECK (COMMRATE IN (0.015, 0.020, 0.025)),
GENDER  CHAR(1) CHECK (GENDER in ( 'M','F')),
STARTDATE  DATE,
DEPT   CHAR (2)
)
```

A view is used as a way of
a) Providing security by limiting what a user can see
- Permission is given to the user to see the view and not the tables

b) Provides a way of simplifying complicated code. The example below is NOT complicated. The complicated code comes next semester.

Create a view of employees in DEPT called AC

```
CREATE VIEW DEPT_AC
AS      SELECT      EMPID, FNAME, LNAME, SALARY, STARTDATE
        FROM        EMPLOYEE
        WHERE       DEPT = 'AC';
```

We can now pretend there exists a table called DEPT_AC with 4 columns. There isn't actually a table, just a structure. Every time you run the view the latest data is inserted into a temporary table and displayed to you.

```
SELECT      EMPID
FROM        DEPT_AC; will get all employees in department AC.
```

Create view and join

```
CREATE TABLE DEPTLIST
AS      SELECT      EMPID, FNAME, LNAME, D.DEPTNO, DNAME
FROM                EMPLOYEE    E, DEPT D
WHERE               EMPLOYEE.DEPTNO = DEPT.DEPTNO;
```

Use a view

```
SELECT      *
FROM        DEPTLIST
WHERE       EMPID > 300;
```

Normalization Review

UNF:

1. Table with repeating group.
2. Identify all the attributes and write them in [], no need to give it a name.
3. The repeating group should be written in ()

The relation identified in 1NF, 2NF and further stages should be named.

1NF:

1. Two dimensional table format.
2. No repeating groups – each row/column intersection only contains one value.
3. Primary key is identified.

2NF:

1. The relation should be in 1NF.
2. No partial dependencies; all non-key columns are fully dependent on the entire primary key.

3NF:

1. The relation should be in 2NF.
2. A non-key column cannot determine the value of another non-key column. Every non-key column must depend directly on the primary key.

4NF:

1. The relation should be in 3NF.
2. It should not have multivalued dependencies. A multivalued fact is one in which several values for a column might be determined by one value for another column.

BCNF:

1. Every determinant in a table is a candidate key. If there is only one candidate key, 3NF and BCNF are the same.

Normalization –Step By Step