Lab 3
Data Definition and Data Manipulation

CSE 4308

DATABASE MANAGEMENT SYSTEMS LAB

AUGUST 24, 2023

1 Data Definition

1.1 Composite Primary key & Foreign key

When a single attribute can not identify the tuples of a table, we have to assign multiple attributes as a single primary key. A composite primary key is a set of columns that uniquely identifies each row in a table. On the other hand, foreign keys are used to restrict the domain of columns of one table to the values of another table. The statement for declaring a composite primary key and foreign key is as follows:

```
CREATE TABLE table_name
(
attribute1 datatype [ NULL | NOT NULL ],
attribute2 datatype [ NULL | NOT NULL ],
...,
[CONSTRAINT constraint_name] PRIMARY KEY (primary_attribute1, ...),
[CONSTRAINT constraint_name] FOREIGN KEY (foreign_attribute1, ...)
REFERENCES reference_table_name
            [ON DELETE CASCADE | SET NULL | SET DEFAULT]
);
```

For example, the DEPT_NAME column of the COURSE table can only have values of the departments that are available in the DEPARTMENT table.

```
CREATE TABLE DEPARTMENT
(
  DEPT_NAME VARCHAR2(20),
  TITLE VARCHAR2(30),
  EST_YEAR VARCHAR2(4),
  CONSTRAINT PK_DEPARTMENT PRIMARY KEY(DEPT_NAME)
);
```

```
CREATE TABLE COURSE
(
  COURSE_ID VARCHAR2(8),
  TITLE VARCHAR2(30),
  PROGRAM VARCHAR2(5),
  DEPT_NAME VARCHAR2(20),
  CREDITS NUMBER,
  CONSTRAINT PK_COURSE PRIMARY KEY(COURSE_ID, PROGRAM),
  CONSTRAINT FK_COURSE_DEPARTMENT FOREIGN KEY(DEPT_NAME)
   REFERENCES DEPARTMENT(DEPT_NAME) ON DELETE CASCADE
);
```

Here you need to ensure the referenced attribute of referencing table should be of the same data type as referenced attribute of the referenced table. It is preferred to use the primary key (or composite primary key) of the referenced table as the foreign key. **Moreover, the referencing table must be created after the referenced table.**
In some cases, a referencing table may need to reference itself, this is the concept of self-referencing.

Interesting Fact

We can also create table by copying the schema and data from another table:

```
CREATE TABLE new_table_name AS
(
SELECT * FROM old_table_name
);
```

1.2 Altering Table Constraint

To add new constraints to the table, we use:

```
ALTER TABLE table_name ADD (CONSTRAINT constraint_name_1 constraint_1,
 CONSTRAINT constraint_name_2 constraint_2,...);
```

```
ALTER TABLE COURSE ADD CONSTRAINT CK_CREDIT CHECK(CEDIT>0.75);
```

To delete constraints from a table, we use:

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name_1,....;
```

```
ALTER TABLE COURSE DROP CONSTRAINT CK_CREDIT;
```

To rename a constraint, we use:

```
ALTER TABLE table_name RENAME CONSTRAINT old_constraint_name
 TO new_constraint_name;
```

```
ALTER TABLE COURSE RENAME CONSTRAINT CK_CREDIT TO CRDT_CHK;
```

Unlike modifying columns, we can not modify any existing constraint in Oracle because of safety concerns.

2 Data Manipulation

2.1 Data Retrieval from Multiple Tables

Remember the basic query structure looks like:

```
SELECT a1, a2, ..., an
FROM r1, r2, ..., rm
WHERE p;
```

We can perform Cartesian Product between multiple tables:

```
SELECT *
  FROM COURSE, DEPARTMENT;
```

Performing Cartesian Product among multiple tables will result in all pair combinations of the tuples of different tables which may contain meaningless records. To overcome this we often use Natural Join. Natural Join ensures that after Cartesian product is performed, only records that have common attribute values between them are kept while the rest are discarded. We can perform Natural Join following this :

```
SELECT *
  FROM COURSE NATURAL JOIN DEPARTMENT;
```

But in the case of Natural join, we have to ensure that both tables share identical column name based on which the tables will be joined.
Alternatively, we can apply conditions to get similar results of Natural Join using the Cartesian Product:

```
SELECT *
  FROM COURSE, DEPARTMENT
        WHERE COURSE.DEPT_NAME=DEPARTMENT.DEPT_NAME;
```

2.2 Upadate Information of the Table

The UPDATE statement is used to modify the existing records in a table.

```
UPDATE table_name
SET attribute1 = value1, attribute2 = value2, ...
WHERE condition;
```

For example,

```
UPDATE COURSE
SET title = 'Database Management Systems', Credit= 3
WHERE COURSE_ID = 'CSE4307';
```

Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

2.3 Delete Information of the Table

The DELETE statement is used to delete existing records in a table.

```
DELETE FROM table_name WHERE condition;
```

For example,

```
DELETE FROM COURSE
WHERE COURSE_ID = 'CSE4307';
```

Similar to the UPDATE statement, not specifying a WHERE clause will result in deleting all the rows.

Tip & Tricks

- To see the list of all tables of a certain user:

  ```
  SELECT table_name FROM user_tables where user=<user_name>;
  ```

- To see the list of users you can use any of these queries:

  ```
  SELECT USERNAME FROM ALL_USERS;

  SELECT USERNAME FROM DBA_USERS;
  ```

- To see the information of current user:

  ```
  SELECT * FROM user_users;
  ```

3 Lab Task

You have to write all SQL statements in an editor first and save them with .sql extension. Then execute the SQL script.

1. Write SQL statements to create the following tables with the given specifications:

   (a) DOCTOR

NAME	VARCHAR2(20)	(e.g.: MR. X, MR. Y)
SPECIALIZATION	CHAR(2)	(e.g. CS, GS, IM, ER)
FEE	NUMBER	(e.g. 2000, 1500)
		Primary Key(NAME, SPECIALIZATION)

   (b) PATIENT

PATIENT_NO	CHAR(5)	(e.g.: P-101) Primary Key
NAME	VARCHAR2(20)	(e.g.: A, B, C) Not Null
ADDRESS	VARCHAR2(10)	(e.g.: DHK, KHL, etc.)

   (c) APPOINTMENT

PATIENT_NO	CHAR(5)	(e.g.: P-101)
NAME	VARCHAR2(20)	(e.g.: MR. X, MR. Y)
SPECIALIZATION	CHAR(2)	(e.g. CS, GS, IM, ER)
		Primary Key(PATIENT_NO, NAME,SPECIALIZATION)

2. Write SQL statements to perform the following alteration operations:

   (a) Add a new attribute 'APPOINTMENT_DATE' (DATE type) in APPOINTMENT table.

   (b) Modify the PRIMARY KEY of APPOINTMENT table and add APPOINTMENT_DATE too with the previous ones.

   (c) Rename the attribute PATIENT_NO, NAME from APPOINTMENT table to P_NO and D_NAME respectively.

   (d) Rename the table APPOINTMENT to APPOINTMENT_INFO.

   (e) Add two foreign key constraints FK_APPOINTMENT_DOCTOR and FK_APPOINTMENT_PATIENT that identifies D_NAME,SPECIALIZATION and P_NO as foreign keys.

3. Insert at least 3 records in each table following the example.

4. Write SQL statements to answer the following queries:

   (a) Find all the Doctors' names whose fees are less than 1500.

   (b) Find all the Patients' names who live in 'KHL' city.

   (c) Show the result of Cartesian Product between PATIENT and APPOINTMENT_INFO table.

   (d) Show the result of Natural Join between PATIENT and APPOINTMENT_INFO table.

(e) Find all the Patient's names and their address who have an appointment today.

(f) Find all the Doctor-related information who have patients from 'DHK'.

(g) Find all Patient-related information who has an appointment with a doctor of 'GS' specialization or a doctor whose fee is greater than 1500.

5. Write following DML statements:

   (a) Update the `NAME` and `ADDRESS` of a tuple from 'A' and 'DHK ' to 'K' and 'RAJ' accordingly.

   (b) Update the `NAME` of table `DOCTOR` from 'MR. Y' as 'Ms. Y'.

   (c) Delete Patient with `PATIENT_NO` P-101.

   (d) Delete all the information without deleting the table structure.