Database Systems
Lab 3
Key Constraints
Version 1.0

The Tables

There will be 3 tables in this practical session- Employee, Department and Dependent. The scenario is about an organization where records about their employees are kept by these 3 tables. The ER diagram is as follows.

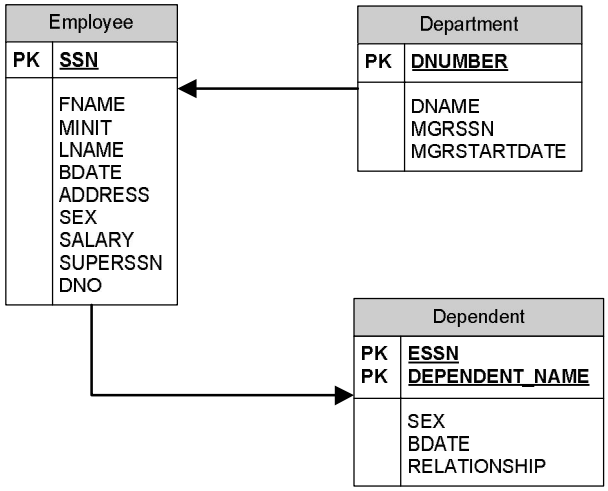

Figure: Schema Diagram of the 3 Tables

Primary Key and Foreign Key

Every table in a database should have a primary key. Primary key is the column or group of columns in a table that uniquely identify every row of that table. Employee table has SSN as its primary key and Department table has DNUMBER as its primary key. If you take a look at the Dependent table, then you will find out two important aspects-

1. Primary key can be made of more than one columns
2. One table can have primary key that is actually a column of another table

The DNO of Employee table is actually the DNUMBER column of Department table. ESSN of Dependent table is the SSN column of Employee table. So, DNO is the foreign key of Employee table and ESSN is the foreign key of Dependent table.

You can declare primary key and foreign key during the creation of table of after the creation of table. Remember one thing- primary keys cannot be NULL.

1. Keys during the Creation of Table

```
CREATE TABLE table_name(
       column_name1 datatype      NOT NULL,
       column_name2 datatype,
       …….
       column_namen datatype,
       PRIMARY KEY (column_name1),
       FOREIGN KEY (column_name2) REFERENCES reference_table_name
);
```

Here, column_name2 is actually a column of reference_table_name.

2. Keys after the Creation of Table

In this case, you can first build up the entire table and then can use ALTER TABLE command to declare primary key and/ or foreign key for that table.

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name
       PRIMARY KEY (column_name1);
ALTER TABLE table_name ADD CONSTRAINT constraint_name
       FOREIGN    KEY    (column_name2)    REFERENCES    reference_table_name
       (column_name_in_reference_table);
```

Demonstration

Now, run the script provided this week to you named **employee.sql**. Just input the command **start drive:/employee.sql**. The script creates the tables, key constraints and the data that the tables contain.

1. Try to delete Employee table by **"DROP TABLE employee;"** command. What does oracle say? As it contains a foreign key which is the primary key of Department table, it won't allow you to drop Employee table.

2. Try to delete Department table as well by **"DROP TABLE department;"** command. Again, oracle won't allow you to delete it.

3. **The only table you can delete among the 3 tables is Dependent table.**

From the schema diagram, you can understand easily why this is happening. **In a "Many to One" relation the table that is "Many side" only, can be dropped**. **But the table that is "one side" of a relation cannot be deleted.**

So, if you want to delete all the tables, you need to drop dependent first, then employee and then department.

Now, take a look at the first two tables- Department and Employee.

```
SELECT * FROM department;
SELECT fname, lname, dno FROM employee;
```

You can see there are 4 entries in Employee table with department number 5. Try to execute following statements and observe what oracle is saying.

```
DELETE FROM department WHERE dnumber=5;
DELETE FROM employee WHERE dno=5;
```

Again take a look at the last two tables- Employee and Dependent.

```
SELECT ssn, fname, lname FROM employee;
SELECT essn, dependent_name FROM dependent;
```

You can see that employee ssn 123456789 has 3 dependents from dependent table and from employee table, you can see that employee's name is **John Smith.**

Now, try to execute the following statement and see what oracle is saying.

```
DELETE FROM employee WHERE ssn='123456789';
```

Now, run the script **employee2.sql.** Again try the following statements to be executed.

```
DELETE FROM department WHERE dnumber=5;
SELECT fname, lname, dno FROM employee;
SELECT * FROM dependent;
```

What do you see? It was impossible with the script **employee.sql.** We have deleted department number 5 from Department table, and then it deleted all the records from Employee and Dependent table that has department number 5 AUTOMATICALLY!

Only one thing is changed in our new script **employee2.sql**. ON DELETE CASCADE statement in the foreign key declaration. You have to take a look at that simple difference. It means- **if you delete data on the master table, all the related entries in detail table will be deleted automatically.**

You now have the facility to delete from the table on many side as well. Execute the following statement.

```
DELETE FROM employee WHERE dno=4;
```

UNIQUE Key

UNIQUE keys are keys that are needed to be unique (not necessarily primary key) throughout the table. For example, if you have a table contains courses for an engineering department, then the name of the subjects do not need to be primary key but they need to be unique. The reason is- you will never ever should find two courses that have the same name.

Run the script named **"UNIQUEKEY.sql".** Take a look at the effect. It has UNIQUE identifier on course names. But one course name is attempted twice to insert. Take a look at the error message while running the script.

CHECK and DEFAULT constraints

Now, we will take a look at these two constraints. On the course table, say all of your courses' pass marks are 40. Then why will you insert 40 in the passmark column every time? DEFAULT can save your time.

Moreover, you may sometimes have to check if right data are inserted or not. What if anyone inserts a course with credit 5? (this is inappropriate for KUET!!). So, you need to check that.

Run the script named "CHECKDEFAULT.sql". See how these two constraints are applied. Then try to violate these by **inserting inappropriate data (e.g., credit hours more than 4 or less than or equal to 0**. See what happens.

Finally

For other group, please drop all the tables-

```
DROP TABLE DEPENDENT;

DROP TABLE EMPLOYEE;

DROP TABLE DEPARTMENT;
```