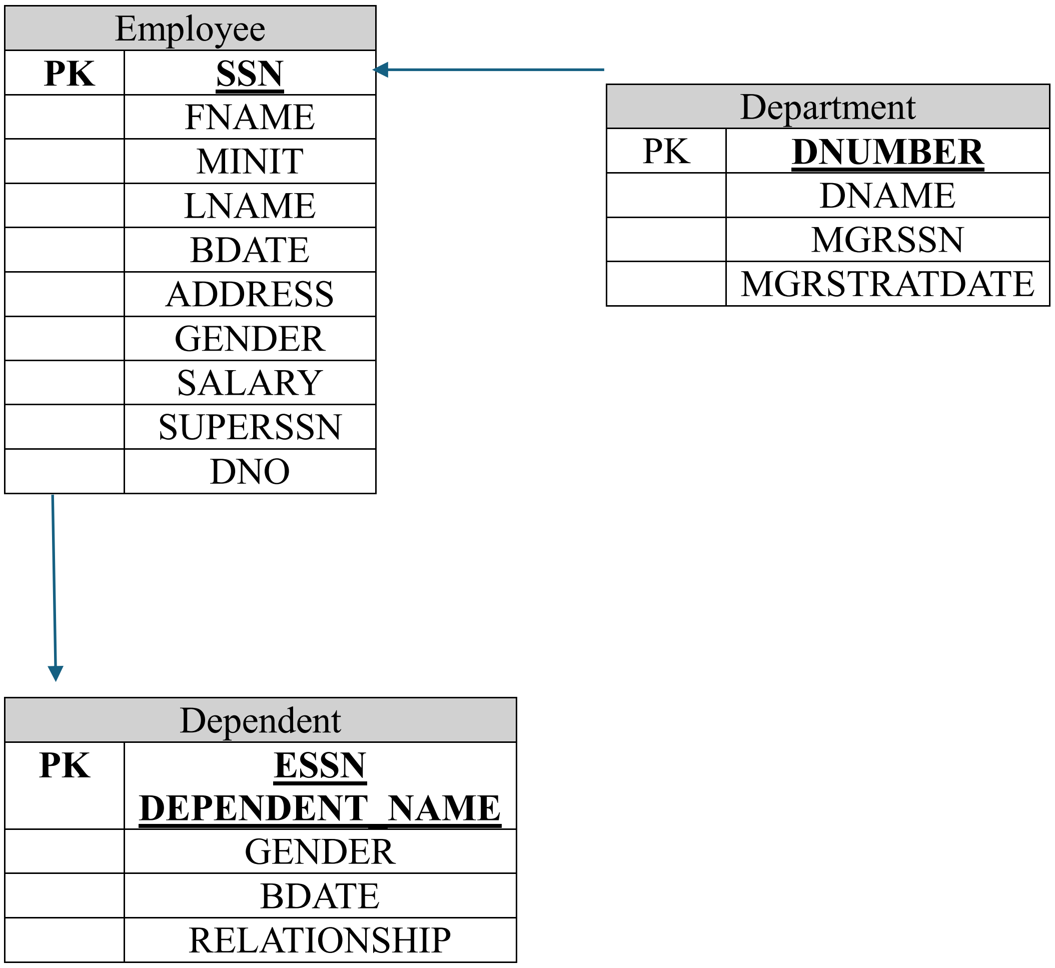
Lab 3

MySQL DML Commands

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



In this database schema, the attributes in each table represent specific details about departments, employees, and their dependents. Here’s a breakdown of each attribute’s role:

DEPARTMENT Table

- **dname**: The name of the department (e.g., "RESEARCH").
- **dnumber**: A unique identifier (primary key) for each department. This ensures each department has a distinct number.
- **mgrssn**: The Social Security Number (SSN) of the manager for that department. This connects the department to the manager in the employee table.
- **mgrstartdate**: The date the manager started managing this department.

EMPLOYEE Table

- **fname**: The first name of the employee.
- **minit**: The middle initial of the employee.
- **lname**: The last name of the employee.
- **ssn**: The Social Security Number of the employee, serving as the primary key. It uniquely identifies each employee.
- **bdate**: The birthdate of the employee.
- **address**: The address of the employee.
- **gender**: The gender of the employee, represented by a single character (M for male, F for female).
- **salary**: The salary of the employee.
- **superssn**: The SSN of the employee’s supervisor. This creates a hierarchical link within the employee table.
- **dno**: The department number (foreign key) indicating which department the employee works in, linking to the dnumber in the department table.

DEPENDENT Table

- **essn**: The SSN of the employee who has the dependent. This foreign key links the dependent to the employee in the employee table.
- **dependent_name**: The name of the dependent, which, combined with essn, serves as part of the primary key, ensuring unique dependents per employee.
- **gender**: The gender of the dependent (M for male, F for female).
- **bdate**: The birthdate of the dependent.
- **relationship**: The relationship of the dependent to the employee (e.g., "SON," "DAUGHTER," "SPOUSE").

Let's create the tables!

```
CREATE TABLE department (
  dname         VARCHAR(15),
  dnumber        INT NOT NULL,
  mgrssn        INT,
  mgrstartdate   DATE,
  PRIMARY KEY (dnumber)
);
```

```
CREATE TABLE employee (
  fname          VARCHAR(15),
  minit         VARCHAR(2),
  lname          VARCHAR(15),
  ssn          INT(12) NOT NULL,
  bdate         DATE,
  address        VARCHAR(35),
  gender         VARCHAR(1),
  salary         INT(7) NOT NULL,
  superssn        INT(12),
  dno           INT NOT NULL,
  PRIMARY KEY (ssn),
  CONSTRAINT fk_dno_dnumber FOREIGN KEY (dno) REFERENCES
department (dnumber)
);

CREATE TABLE dependent (
  essn               INT,
  dependent_name        VARCHAR(15),
  gender               VARCHAR(1),
  bdate               DATE,
  relationship          VARCHAR(12),
  PRIMARY KEY (essn, dependent_name),
  CONSTRAINT fk_essn_ssn FOREIGN KEY (essn) REFERENCES employee
(ssn)
);
```

Data Insertion into a Table

Syntax:

INSERT INTO <table name> (attribute1, attribute2, ...)

VALUES (<value for attribute1>, <value for attribute2>, ...)

Example:

Insert into department (dept_name, building, budget)

values ('CSE', 'Main Campus', 1000000)

*** If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

INSERT INTO table_name

VALUES (value1, value2, value3, ...);

Example:

Insert into department values ('CSE', 'Main Campus', 1000000)

Let’s insert some data into our tables!

```
-- Insert values into department table

INSERT INTO department VALUES ('RESEARCH', 5, 333445555, '1978-05-22');

INSERT INTO department VALUES ('ADMINISTRATION', 4, 987654321, '1985-01-01');

INSERT INTO department VALUES ('HEADQUARTERS', 1, 888665555, '1971-06-19');


-- Insert values into employee table

INSERT INTO employee VALUES

('JOHN','B','SMITH',123456789,'1955-01-09','731 FONDREN, HOUSTON, TX', 'M',30000,333445555,5);

INSERT INTO employee VALUES

('FRANKLIN','T','WONG',333445555,'1945-12-08','638 VOSS, HOUSTON, TX', 'M',40000,888665555,5);

INSERT INTO employee VALUES

('ALICIA','J','ZELAYA',999887777,'1958-07-19','3321 CASTLE, SPRING, TX', 'F',25000,987654321,4);

INSERT INTO employee VALUES

('JENNIFER','S','WALLACE',987654321,'1931-06-20','291 BERRY, BELLAIRE, TX', 'F',43000,888665555,4);

INSERT INTO employee VALUES

('RAMESH','K','NARAYAN',666884444,'1952-09-15','975 FIRE OAK, HUMBLE, TX', 'M',38000,333445555,5);

INSERT INTO employee VALUES

('JOYCE','A','ENGLISH',453453453,'1962-07-31','5631 RICE, HOUSTON, TX', 'F',25000,333445555,5);
```

INSERT INTO employee VALUES

('AHMAD','V','JABBAR',987987987,'1959-03-29','980 DALLAS, HOUSTON, TX', 'M',25000,987654321,4);

INSERT INTO employee VALUES

('JAMES','E','BORG',888665555,'1927-11-10', '450 STONE, HOUSTON, TX', 'M',55000,NULL,1);

-- Insert values into dependent table

INSERT INTO dependent VALUES (333445555,'ALICE','F','1976-04-05','DAUGHTER');

INSERT INTO dependent VALUES (333445555,'THEODORE','M','1973-10-25','SON');

INSERT INTO dependent VALUES (333445555,'JOY','F','1948-05-03','SPOUSE');

INSERT INTO dependent VALUES (123456789,'MICHAEL','M','1978-01-01','SON');

INSERT INTO dependent VALUES (123456789,'ALICE','F','1978-12-31','DAUGHTER');

INSERT INTO dependent VALUES (123456789,'ELIZABETH','F','1957-05-05','SPOUSE');

INSERT INTO dependent VALUES (987654321,'ABNER','M','1932-02-26','SPOUSE');

Now

1. Try to delete Employee table by “DROP TABLE employee;” command. What does

mySQL 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, it won't allow you to delete it.

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

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

Retrieving/Searching some Data/rows/records/results from database table (SQL query)

Syntax:

SELECT <column_name_needs_to_be_shown>

FROM <table_name>

WHERE <some_condition>

Example:

• Ques: Show firstName, lastName, age from the students table whose student_id is 1163172

Ans:

SELECT firstName, lastName, age

FROM students

WHERE student_id = 1163172

• Ques: Show everything (or every detail) of the employee named 'John Doe' from the employee table

Ans: (Hint   * means everything/all the columns)

SELECT *

FROM employee

WHERE employee_name = 'John Doe' – (remember, if it's a string we have to put it under single quotations)

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

SELECT * FROM department;

SELECT fname, lname, dno FROM employee;

SELECT fname, lname, dno

FROM employee

WHERE dno=5;

Data Modification in a Table

Syntax:

UPDATE <table name>

SET <attribute name> = <new value>

WHERE <someCondition_on_column_values>;

Example:

Update department

set budget = 1500000

where dept_name = 'CSE';

Note: Be careful when updating records in a table! Notice the `WHERE` clause in the `UPDATE` statement. The `WHERE` clause specifies which record(s) that should be updated. If you omit the `WHERE` clause, all records in the table will be updated!

Suppose you want to update the salary of an employee with the ssn of 123456789 to a new salary of 35000:

```
UPDATE employee

SET salary = 35000

WHERE ssn = 123456789;
```

If you want to update the manager's SSN (`mgrssn`) in the `department` table for the `RESEARCH` department to a new manager with SSN `987987987`:

```
UPDATE department
SET mgrssn = 987987987
WHERE dname = 'RESEARCH';
```

Data Deletion from a Table

Syntax:

DELETE FROM <table name>

WHERE <someCondition>;

Example:

Delete from department

Where budget<10000;

Note: Be careful when deleting records in a table! Notice the `WHERE` clause in the `DELETE` statement. The `WHERE` clause specifies which record(s) should be deleted. If you omit the `WHERE` clause, all records in the table will be deleted!

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 .

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

Note: Deleting an employee may affect referential integrity if there are records in the `dependent` table that reference the employee's `ssn`. You may need to delete or update the related records in `dependent` before deleting the employee, or use `ON DELETE CASCADE` in the foreign key constraint when creating the table.

It means- if you

delete data on the master table, all the related entries in detail table will be deleted automatically.

```
CREATE TABLE dependent (
  essn                  INT(12),
  dependent_name        VARCHAR(15),
  gender                VARCHAR(1),
  bdate                 DATE,
  relationship          VARCHAR(12),
  PRIMARY KEY (essn, dependent_name),
  FOREIGN KEY (essn) REFERENCES employee (ssn) ON DELETE CASCADE
);
```

With ON DELETE CASCADE, when an employee is deleted from the employee table, all dependents associated with that employee (based on essn) will automatically be deleted from the dependent table.