Omar Ahmadi HW 1 1.6, 2.2, 2.4, 3.8, 3.14

Exercise 1.6

Scrooge McNugget wants to store information (names, addresses, descriptions of embarrassing moments, etc.) about the many ducks on his payroll. Not surprisingly, the volume of data compels him to buy a database system. To save money, he wants to buy one with the fewest possible features, and he plans to run it as a stand-alone application on his PC clone. Of course, Scrooge does not plan to share his list with anyone. Indicate which of the following DBMS features Scrooge should pay for; in each case, also indicate why Scrooge should (or should not) pay for that feature in the system he buys.

1. A security facility is needed in order to prevent another user from querying Scrooge's Database.

2. Concurrency control isn't needed because only Scrooge queries the database.

3. Crash Recovery is always needed in order to save his data in case of a power outage.

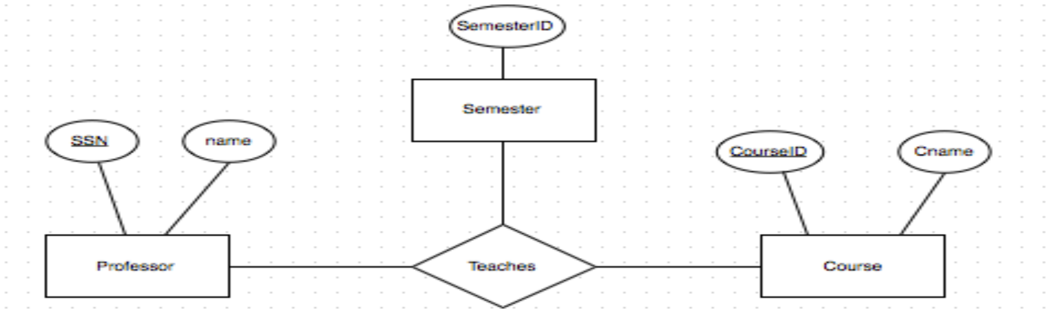
4. A view mechanism is needed for convenience and to avoid writing the same queries repeatedly

5. A query language is necessary to analyze data and define views.

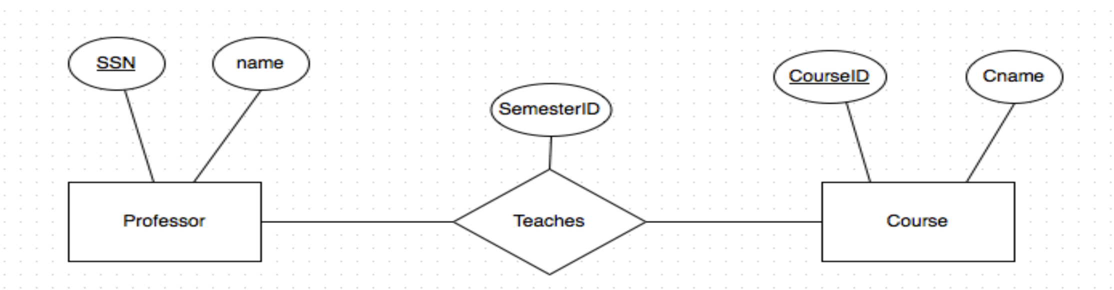
Exercise 2.2

A university database contains information about professors (identified by social security number, or SSN) and courses (identified by courseid). Professors teach courses; each of the following situations concerns the Teaches relationship set. For each situation, draw an ER diagram that describes it (assuming no further constraints hold).

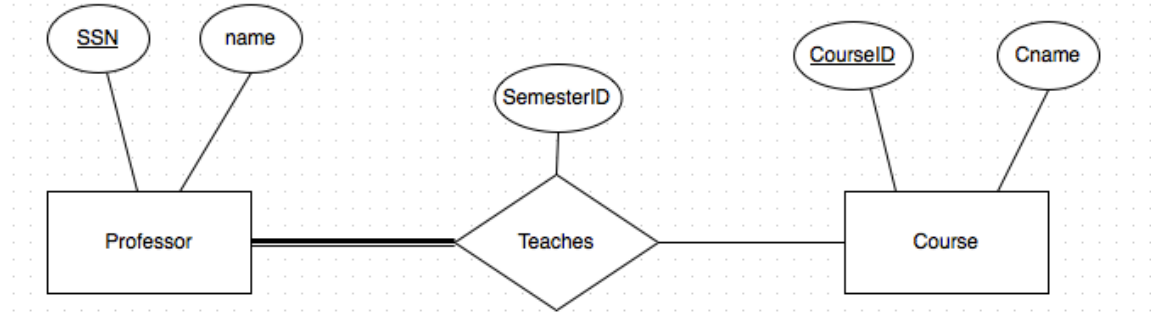
1. Professors can teach the same course in several semesters, and each offering must be recorded.



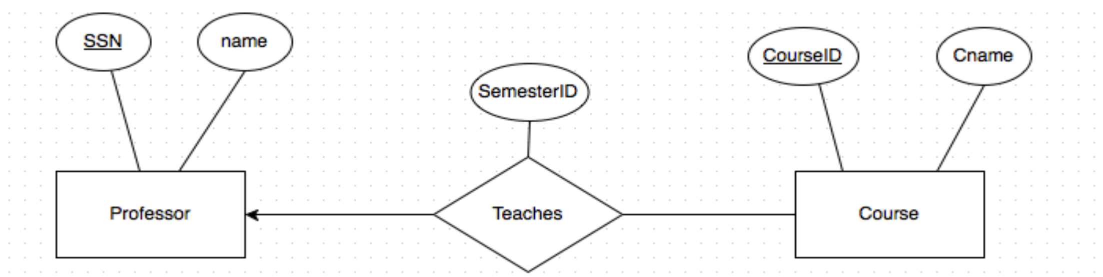
2. Professors can teach the same course in several semesters, and only the most recent such offering needs to be recorded. (Assume this condition applies in all subsequent questions.)



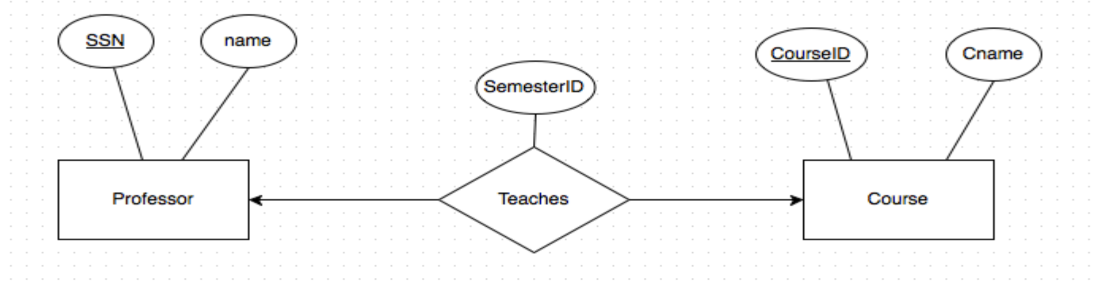
3. Every professor must teach some course.



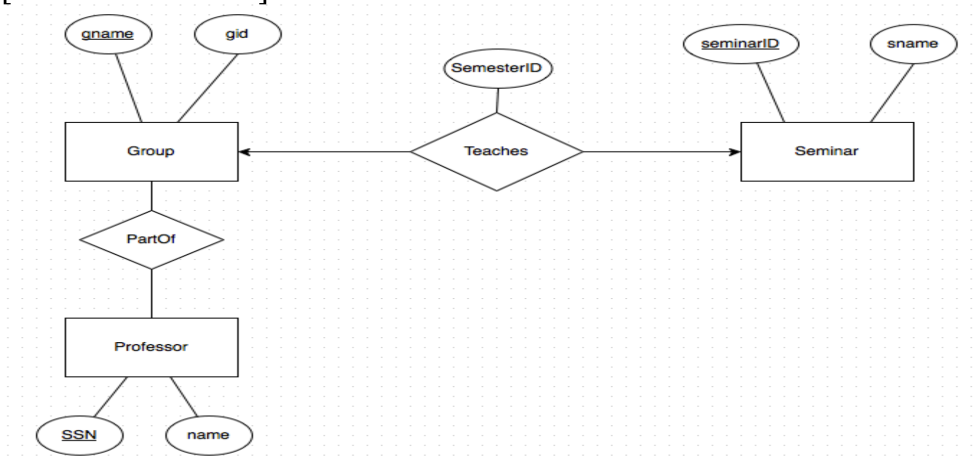
4. Every professor teaches exactly one course (no more, no less).[REVERSE ARROW]



5. Every professor teaches exactly one course (no more, no less), and every course must be taught by some professor. [REVERSE ARROW]

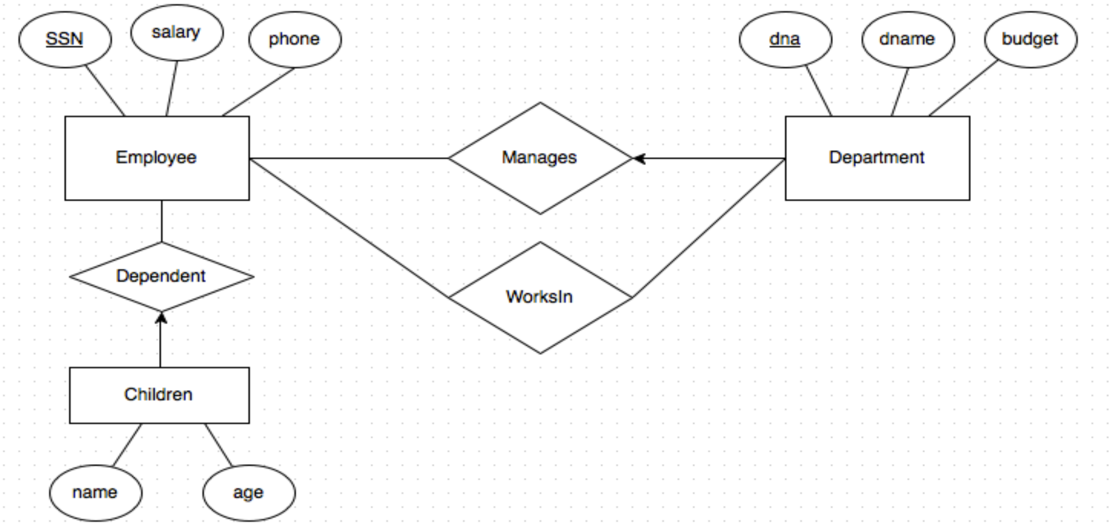


6. Now suppose that certain courses can be taught by a team of professors jointly, but it is possible that no one professor in a team can teach the course. Model this situation, introducing additional entity sets and relationship sets if necessary. [REVERSE ARROW]



Exercise 2.4

A company database needs to store information about employees (identified by ssn, with salary and phone as attributes), departments (identified by dna, with dname and budget as attributes), and children of employees (with name and age as attributes). Employees work in departments; each department is managed by an employee; a child must be identified uniquely by name when the parent (who is an employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company. Draw an ER diagram that captures this information.



Exercise 3.8

Emp(eid: integer, ename: string, age: integer, salary: real)
Works(eid: integer, did: integer, pct_time: integer)
Dept(did: integer, dname: string, budget: real, managerid: integer)

1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?

```
CREATE TABLE Works{
        eid INTEGER,
        did INTEGER,
        pct_time INTEGER,
        PRIMARY KEY(eid,did),
        FOREIGN KEY(eid) REFERENCES Emp(eid),
        FOREIGN KEY(did) REFERENCES Dept(did)
}
```

Options: NO ACTION, CASCADE, SET DEFAULT, SET NULL

2. Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints.

```
CREATE TABLE Emp(
        eid INTEGER,
        ename CHAR(50)
        age INTEGER,
        salary REAL,
        PRIMARY KEY(eid) )
CREATE TABLE Works(
        eid INTEGER,
        did INTEGER,
        pct_time INTEGER,
        PRIMARY KEY(eid,did),
        FOREIGN KEY(eid) REFERENCES Emp(eid)
                            ON DELETE    CASCADE,
        FOREIGN KEY(did) REFERENCES Dept(did)
                            ON DELETE    CASCADE,
)
CREATE TABLE Dept(
        did INTEGER,
        dname CHAR(20),
        budget REAL,
        managerid INTEGER,
        PRIMARY KEY(did),
        FOREIGN KEY(managerid) REFERENCES Emp(eid)
                            ON DELETE    SET NULL,
)
```

3. Define the Dept relation in SQL so that every department is guaranteed to have a manager.

```
CREATE TABLE Dept(
        did INTEGER,
        dname CHAR(20),
        budget REAL,
        managerid INTEGER NOT NULL,
        PRIMARY KEY(did),
        FOREIGN KEY(managerid) REFERENCES Emp(eid)
)
```

4. Write an SQL statement to add John Doe as an employee with eid = 101, age = 32 and salary = 15,000.

```
INSERT
INTO EMP
VALUES(101, ‘John Doe’,32,15000)
```

5. Write an SQL statement to give every employee a 10 percent raise.

```
UPDATE Emp E
SET E.salary*1.1
```

6. Write an SQL statement to delete the Toy department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.

```
DELETE
FROM Dept D
WHERE D.dname = ‘Toy’
```

Exercise 3.14

Consider the scenario from Exercise 2.4, where you designed an ER diagram for a company database. Write SQL statements to create the corresponding relations and capture as many of the constraints as possible. If you cannot capture some constraints, explain why.

```
CREATE TABLE (
        ssn CHAR(10),
        salary INTEGER,
        phone CHAR(13),
        PRIMARY KEY (ssn) )
CREATE TABLE Departments (
        dna INTEGER,
        budget INTEGER,
        dname CHAR(20),
        PRIMARY KEY (dna) )
CREATE TABLE WorksIn (
        ssn CHAR(10),
        dna INTEGER,
        PRIMARY KEY (ssn, dna),
        FOREIGN KEY (ssn) REFERENCES Employees,
        FOREIGN KEY (dna) REFERENCES Departments )
```

```
CREATE TABLE Manages (
        ssn CHAR(10),
        dna INTEGER,
        PRIMARY KEY (dna),
        FOREIGN KEY (ssn) REFERENCES Employees,
        FOREIGN KEY (dna) REFERENCES Departments )
CREATE TABLE Dependent (
        ssn CHAR(10),
        name CHAR(10),
        age INTEGER,
        PRIMARY KEY (ssn, name),
        FOREIGN KEY (ssn) REFERENCES Employees,
                                ON DELETE CASCADE )
```