Database Design

(1)

Hogwarts school has decided to step into the modern world and has asked you to design its relational database. It has given the following list of basic conditions that are needed for its database.

• Professors have a Wizard ID which is a 10 digit number, a name, a rank, the number of years they have been in the school and their research specialty.

• Students have an Apprentice ID which is a 10 digit number, a name, the year of study and their gpa.

• Professors teach Courses which are offered by the school. A course has a course ID, a name and the number of credits. Every course is taught by at most one professor and a professor may teach many courses. Courses are taught in terms. The teacher of a course may vary for different terms.

• There are various houses to which students belong to. A student is a member of exactly one house.

• Houses are identified by their house ID and have a name and the number of students in the school who are currently a member of the house. In addition, every house has precisely one student who is the head of the house. Each house name is distinct.

• Students are also enrolled in courses. A student can enroll in many courses in a given term and a course contains many students. The grade of the student in the course that is offered in a term is also to be noted along with the enrollment information and is unique. A student may enroll in the same course in different terms.

• Every student has at most one professor who advises the student. A professor may advise MANY students.

List the various entities and their corresponding attributes in the design. For each relationship, describe the key constraints, type of the relationship (one to many or many to many) and descriptive attributes (if any). You can use ER diagram or textual listing.

Answer:

The entities and their attributes are:

i. Professor: Wizard ID (WID), Number of Years (numYears), Professor Name (pname), Rank (rank) and Specialization (spec)

ii. Student: Apprentice ID (SID), Student Name (sname), Year of Study (year), gpa (gpa)

iii. Course: Course ID (CID), Course Name (cname), Number of Credits (numCred)

iv. House: House ID (HID), House Name (hname), Number of Members (numMem), Head ID (headID)

The relationships are:

i. Teaches: Key constraint on Course - A course can have at most one professor who teaches it, Relationship is <u>one to many</u> - A professor can teach many courses, Descriptive attribute of the relationship is the Term of offering (term)

ii. Advises: Key constraint on advises - A student can have at most one professor who advises the student, Relationship is one to many - A professor can advise many students, There are no descriptive attributes

iii. Enrolled: The relationship is <u>many to many</u>, The Descriptive attributes are grade and term of offering. Additionally there is a foreign key constraint that (cid,term) is a foreign key in the relation Teaches. This models the scenario that a student is enrolled in a course that is taught or offered in a given term. (However, this is constraint is not strictly enforce for grading)

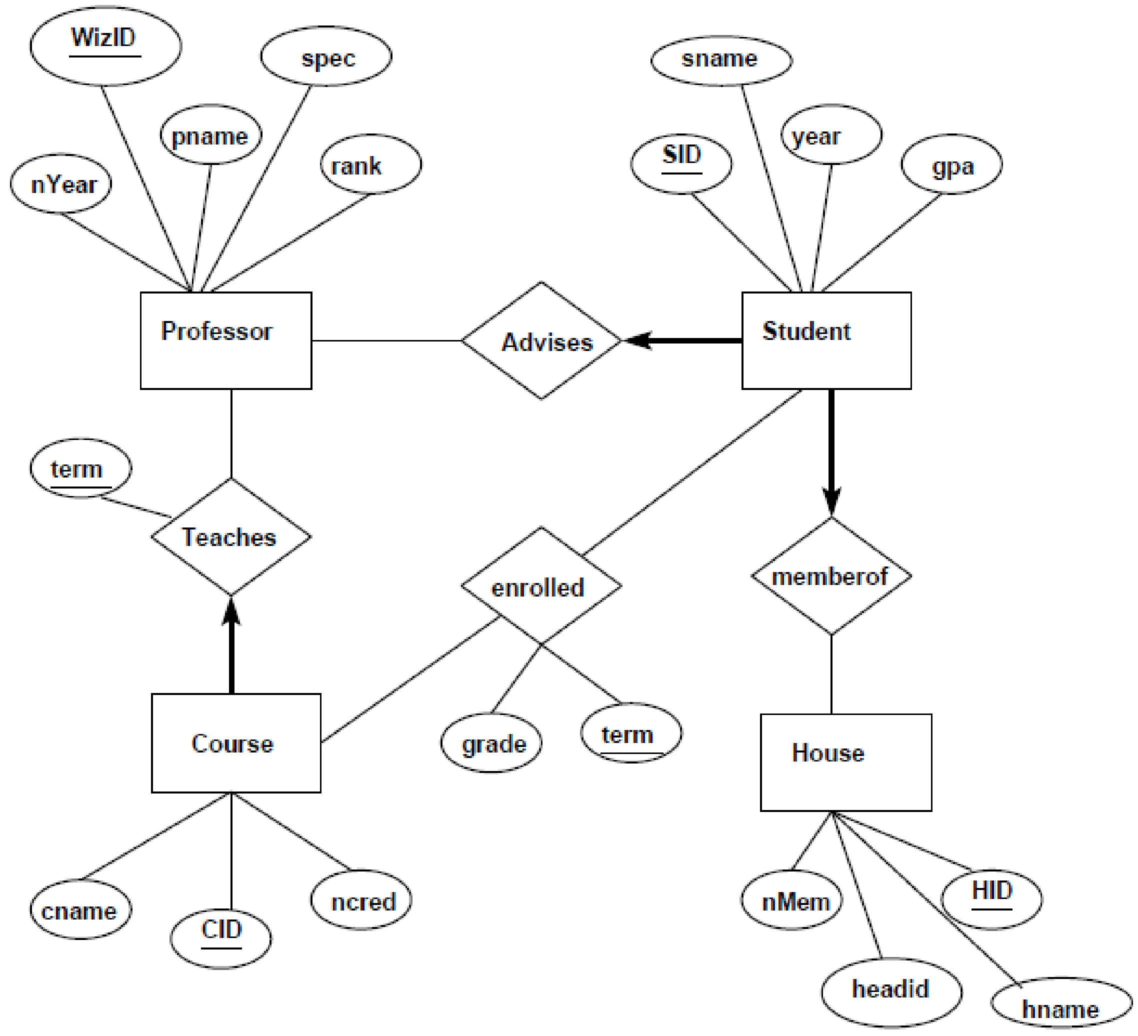
iv. MemberOf: Key constraint on MemberOf - A student can be a member of at most one house, Relationship is <u>one to many</u> - A house can have many students,
There are no descriptive attributes.

(2)

Design and draw an ER diagram that captures the information required by the school. Use entities, relationships and attributes and be sure to indicate the key and participation constraints.

Answer:

The ER diagram is shown below. There are a few points to note. The advises relationship has total participation from students. However, since the question says atmost one professor who advises a student, there is some ambiguity in interpretation here. So an answer where the thick line is absent for the advises relationship from students is also a valid one.

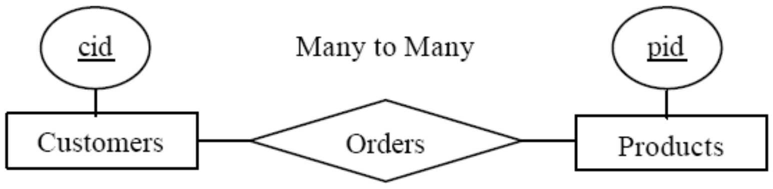


SQL DDL

(3)

Show the SQL-DDL statements that create the tables including the foreign key and primary key indications for the following ER diagrams

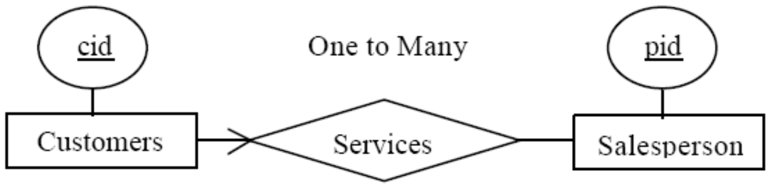
(i)



Answer:

```
CREATE TABLE Customers (
      cid CHAR(10),
      primary key (cid))
CREATE TABLE Products (
      pid CHAR(10),
      primary key (pid))
CREATE TABLE Orders (
      cid CHAR(10),
      pid CHAR(10),
      PRIMARY KEY (cid, pid),
      FOREIGN KEY (cid) REFERENCES Customers,
      FOREIGN KEY (pid) REFERENCES Products)
```

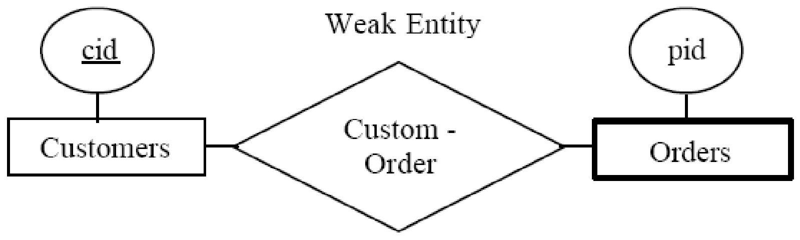
(ii)



Answer:

```
CREATE TABLE Salespersons (
      sid CHAR(10),
      primary key (sid))
CREATE TABLE Customers(
      cid CHAR(10),
      sid CHAR(10),
      PRIMARY KEY (cid),
      FOREIGN KEY (sid) REFERENCES Salespersons)
```

(iii)

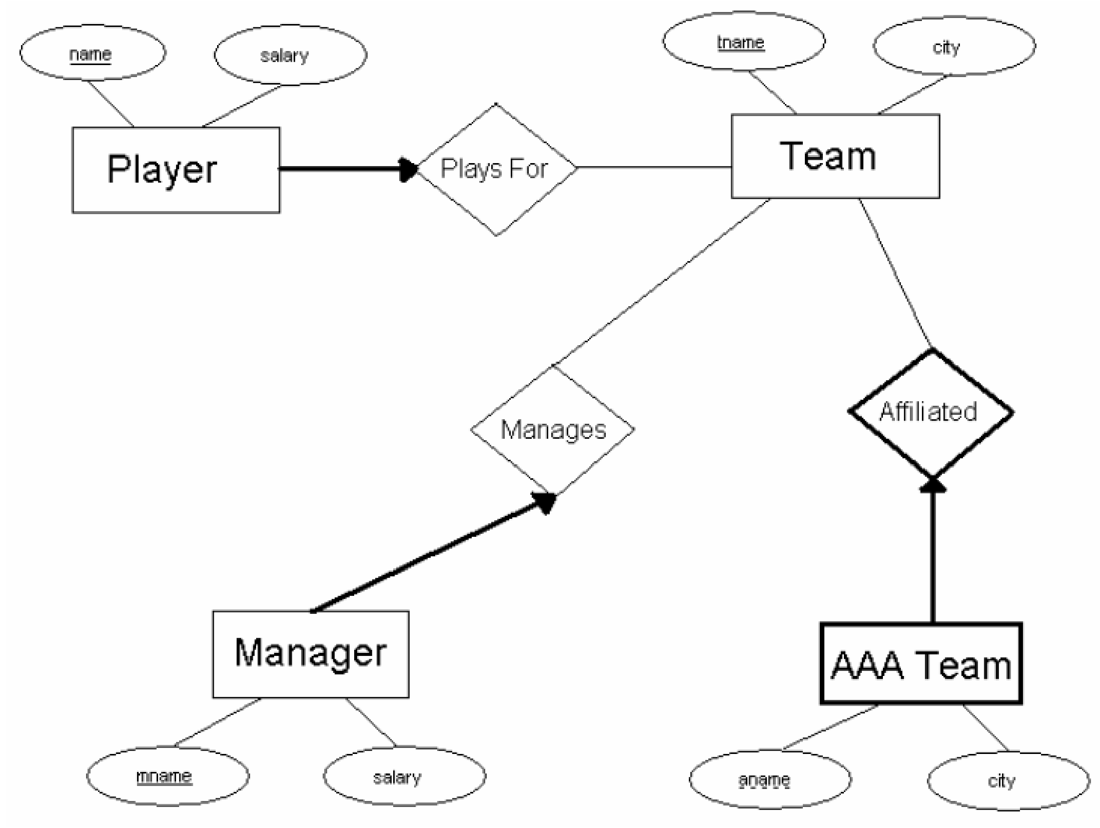


Answer:

```
CREATE TABLE Customers (
      cid CHAR(10),
      primary key (cid))
CREATE TABLE Orders (
      cid CHAR(10),
      pid CHAR(10),
      PRIMARY KEY (cid, pid),
      FOREIGN KEY (cid) REFERENCES Customers)
```

(4)

Using SQL, convert the following ER diagram to the relational model. You do not need CHECK constraints.



Answer:

```
CREATE TABLE Player (name char(30),
                     salary real,
                     tname char(30) NOT NULL,
                     PRIMARY KEY (name),
                     FOREIGN KEY (tname) REFERENCES Team);
CREATE TABLE Team (tname char(30),
                   city char(30),
                   PRIMARY KEY (tname));
CREATE TABLE Manager (mname char(30),
                      salary real,
                      tname char(30) NOT NULL,
                      PRIMARY KEY (mname),
                      FOREIGN KEY (tname) REFERENCES Team);
CREATE TABLE AAATeam (tname char(30),
                      aname char(30),
                      city char(30),
                      PRIMARY KEY (tname, aname),
                      FOREIGN KEY (tname) REFERENCES Team);
```

Relational Algebra

(5)

Consider the following schema.

Suppliers(sid :integer, sname:string, address:string)

Parts(pid:integer, pname:string, color :string)

Catalog(sid :integer, pid:string, cost:real)

The key fields of each relation are underlined. Write the following queries in Relational Algebra. (There may be many ways to answer each query.)

(a) Find the names of suppliers who supply some yellow part.

Answer:

$$\pi_{sname}(Suppliers \bowtie (\sigma_{color='yellow'}(Parts) \bowtie Catalog))$$

(b) Find the sids of suppliers who supply some green part but not a red part.

Answer:

$$\pi_{sid}(Suppliers \bowtie (\sigma_{color='green'}(Parts) \bowtie Catalog)) - \pi_{sid}(Suppliers \bowtie (\sigma_{color='red'}(Parts) \bowtie Catalog))$$

(c) Find the sids and snames of suppliers who supply a ’bolt’ whose price is under 100 dollars or whose color is red.

Answer:

There is ambiguity here. Some of you may interpret the question to mean a bolt whose price is under 100 dollars or a bolt whose color is red. Some of you may interpret it to mean a bolt whose price is under 100 dollars or a red part. I have given points to both these versions.

$$\pi_{sid,sname}(Suppliers \bowtie (\sigma_{(pname='bolt' \wedge cost<100) \vee (color='red')}(Parts \bowtie Catalog)))$$

$$\pi_{sid,sname}(Suppliers \bowtie (\sigma_{(pname='bolt' \wedge (cost<100 \vee color='red'))}(Parts \bowtie Catalog)))$$

(d) Find the sids of suppliers who supply all parts.

Answer:

$$\pi_{sid}((\pi_{sid,pid}(Catalog))/(\pi_{pid}(Parts))$$

(e) Find pids of parts supplied by at least two different suppliers

Answer:

$\rho(CatPairs(1 \rightarrow sid_1, 2 \rightarrow pid_1, 3 \rightarrow cost_1, 4 \rightarrow sid_2, 5 \rightarrow pid_2, 6 \rightarrow cost_2), Catalog \times Catalog)$

$\pi_{pid_1}(\sigma_{(pid_1=pid_2)\wedge(sid_1\neq sid_2)}CatPairs)$

(f) Find the names of suppliers who supply some brown part.

Answer:

$$\pi_{sname}(Suppliers \bowtie (\sigma_{color='brown'}(Parts) \bowtie Catalog))$$

(g) Find the sids of suppliers who supply some yellow part but not a red part.

Answer:

$$\pi_{sid}(Suppliers \bowtie (\sigma_{color='yellow'}(Parts) \bowtie Catalog)) - \pi_{sid}(Suppliers \bowtie (\sigma_{color='red'}(Parts) \bowtie Catalog))$$

(h) Find the sids and snames of suppliers who supply a 'nut' whose price is under 10 dollars or whose color is pink.

Answer:
There is ambiguity here too. Some of you interpret the question as a nut whose price is under 10 dollars or a nut whose color is pink. Some of you interpret it as a nut whose price is under 10 dollars and a part whose color is pink. Both versions are treated as correct answers.

$$\pi_{sid,sname}(Suppliers \bowtie (\sigma_{(pname='nut'\wedge cost<10)\vee(color='pink')}(Parts \bowtie Catalog)))$$

$$\pi_{sid,sname}(Suppliers \bowtie (\sigma_{(pname='nut'\wedge(cost<10\vee color='pink'))}(Parts \bowtie Catalog)))$$

Relational Algebra and SQL

(6)

Given the following database tables, choose all queries in SQL, Relational Algebra, or both. Note that each question may have more than one correct answers. Circle all that apply.

Primary keys are underlined. Note that the driver involved in a car accident may not always be the owner of the car. Assume that accident_date is of type integer, and represents a year (e.g. 1980). Year is also of type integer. We assume that a car cannot get involved in more than one accident at a certain date.

```
Person(SSN, name, address)
Car(license, year, model)
Accident(license, accident_date, driver, damage_amount)
Owns(SSN, license)
```

1. Find the SSN of every person who owns one or more cars, none of which has ever been involved in a car accident.

A.

```
SELECT O.SSN
FROM Owns O
WHERE O.license NOT IN (SELECT A.license FROM Accident A);
```

B.

$$\pi_{SSN}(Owns) - \pi_{SSN}(Owns \bowtie Accident)$$

C.
None of the above

Answer:
B is right. A return the SSN of every person that has at least one car that is not involved in an accident, while the question asks for the people for whom all of their cars have never been involved in an accident.

2. Find the SSN of every person, who owns a TOYOTA or a DODGE

A.

```
SELECT O.SSN
FROM Owns O, Car C
WHERE O.license=C.license AND (C.model=’TOYOTA’ OR
                                                    C.model=’DODGE’)
```

B.

$$\pi_{SSN}\left(Owns \rhd\lhd \sigma_{model='TOYOTA' \vee model='DODGE'}\left(Car\right)\right)$$

C. None of the above

Answer:
A, B are right.

3. Find the SSN of all persons who have had all of their cars involved in an accident.

A.

```
SELECT O.SSN
FROM Owns O, Accident A
WHERE O.license=A.license
        EXCEPT
SELECT O.SSN
FROM Owns O
WHERE O.license NOT IN (SELECT A.license FROM Accident A);
```

B.

$$\pi_{SSN,license}\left(Owns \rhd\lhd Accident\right) \div \pi_{license}\left(Accident\right)$$

C.
None of the above.

Answer:

A is right. B returns the people who have been involved in every accident. To be right it should be

$$\pi_{SSN,license}\left(Owns \rhd\lhd Accident\right) \div \pi_{license}\left(Owns\right)$$

4. Who is the driver who participated in the most costly accident? Return the driver and the amount of damage.

A.
```
SELECT driver, damage_amount
FROM Accident
WHERE damage_amount IN (SELECT MAX(damage_amount)
                                          FROM Accident)
```

B.
```
SELECT driver, damage_amount
FROM Accident
WHERE damage_amount=MAX(damage_amount)
```

C.
```
 (SELECT driver, damage_amount
  FROM Accident)
 EXCEPT
(SELECT a1.driver, a1.damage_amount
 FROM Accident a1, Accident a2
 WHERE a1.damage_amount<a2.damage_amount AND a1.driver<>a2.driver)
```

D.
None of the above.

Answer:
A is right. B doesn't work. C doesn't work if the driver involved in the most expensive accident, has also been involved to some other accident.

5. (4 points) Find the license number of all cars that had an accident in less than 5 years after their production. Return the license number and the number of years

A.
```
SELECT c.license
FROM Car c, Accident a
WHERE a.accident_date-c.year<5
```

B.

$$\pi_{license}\left(\sigma_{accident\_date-year<5}\left(Car \bowtie Accident\right)\right)$$

C.
None of the above

Answer:
C is right. None of the queries return the number of years.