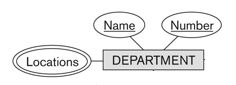
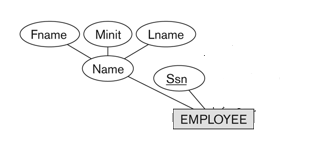
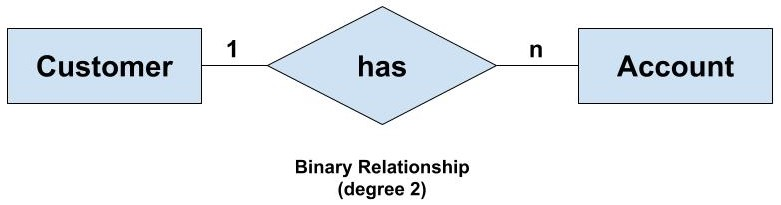
**Mid Exam with solution**

Q1. Give one example of each of the following terms with the help of an Entity Relationship(ER) diagram. [3+3+3+3=12Points]

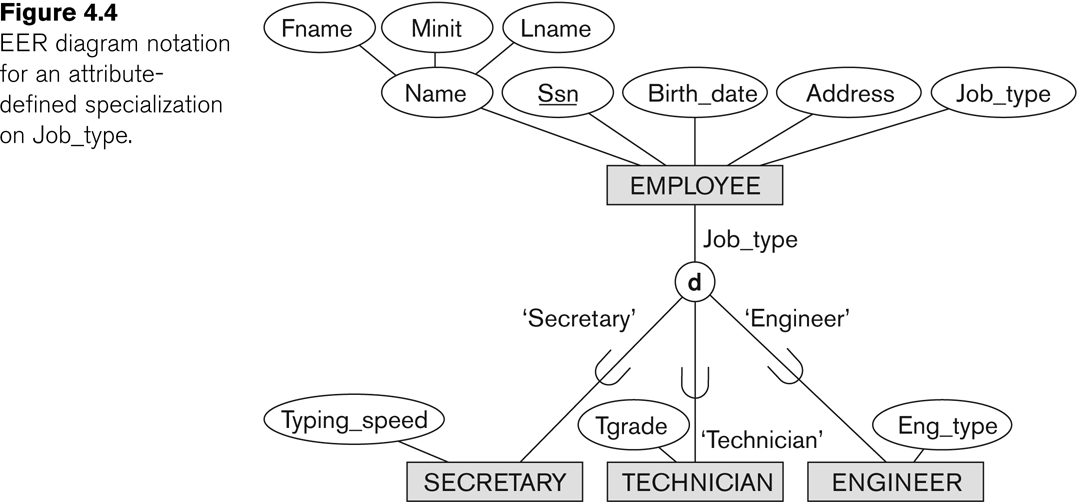
1. Multivalued and composite attribute



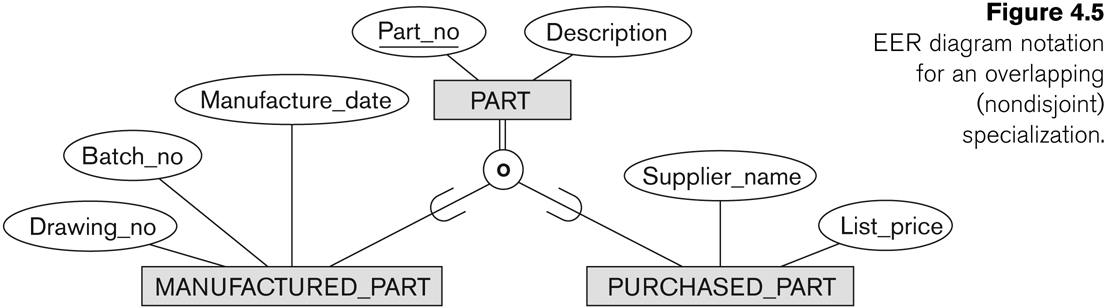
1. Binary one to many relationships



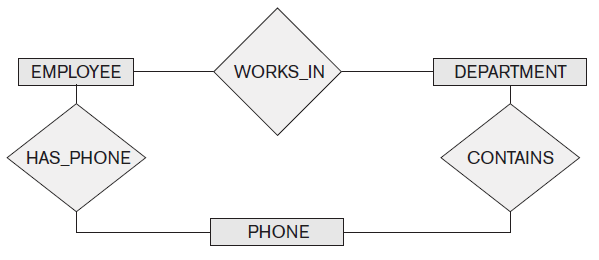
1. Disjoint Partial Specialization



1. Overlapping Total Specialization

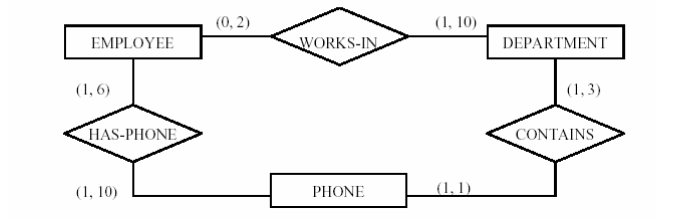


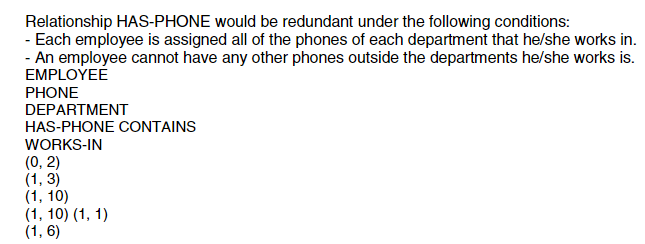
Q2. Consider the following ER diagram [10 Points]



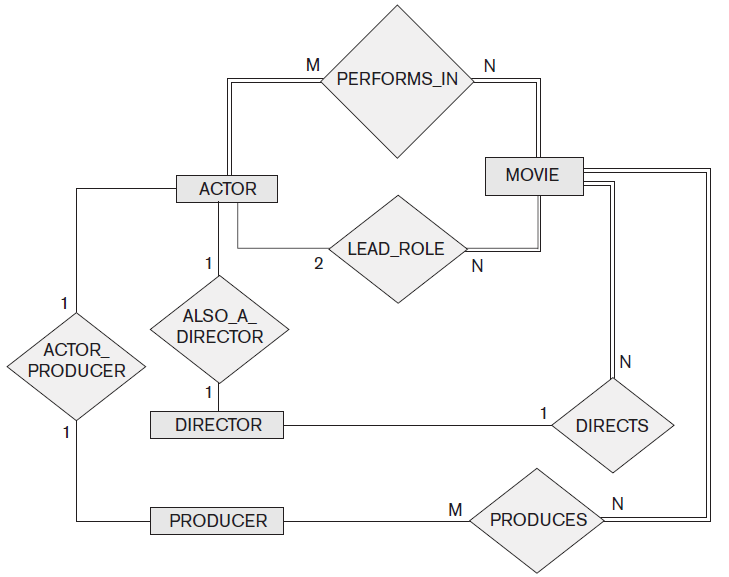
Assume that an employee may work in up to two departments or may not be assigned to any department. Assume that each department must have one and may have up to three phone numbers. Supply (min, max) constraints on this diagram. *State clearly any additional assumptions you make.* Under what conditions would the relationship HAS\_PHONE be redundant in this example?

Solution:





Q3. Consider the ER schema for the MOVIES database in the figure below.

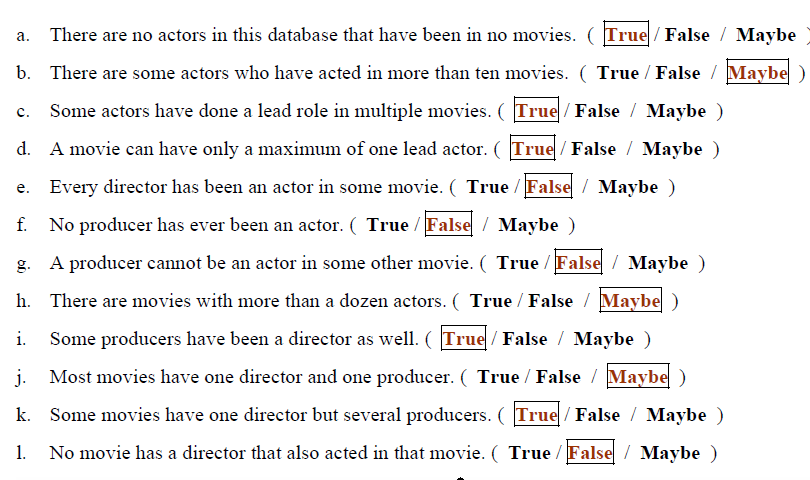


Assume that MOVIES is a populated database. ACTOR is used as a generic term and includes actresses.

Given the constraints shown in the ER schema, respond to the following statements with *True, False*, or *Maybe*. Assign a response of *Maybe* to statements that, although not explicitly shown to be *True*, cannot be proven *False* based on the schema as shown. Encircle your answer [13 points]

|  |  |
| --- | --- |
| **Statement** | ***Answer*** |
| 1. There are no actors in this database that have been in no movies. | *True, False*, or *Maybe* |
| 1. There are some actors who have acted in more than ten movies. | *True, False*, or *Maybe* |
| 1. Some actors have done a lead role in multiple movies. | *True, False*, or *Maybe* |
| 1. A movie can have only a maximum of two lead actors. | *True, False*, or *Maybe* |
| 1. Every director has been an actor in some movie. | *True, False*, or *Maybe* |
| 1. No producer has ever been an actor. | *True, False*, or *Maybe* |
| 1. A producer cannot be an actor in some other movie. | *True, False*, or *Maybe* |
| 1. There are movies with more than a dozen actors. | *True, False*, or *Maybe* |
| 1. Some producers have been a director as well. | *True, False*, or *Maybe* |
| 1. Most movies have one director and one producer. | *True, False*, or *Maybe* |
| 1. Some movies have one director but several producers. | *True, False*, or *Maybe* |
| 1. There are some actors who have done a lead role, directed a movie, and produced a movie | *True, False*, or *Maybe* |
| 1. No movie has a director who also acted in that movie. | *True, False*, or *Maybe* |

Solution:

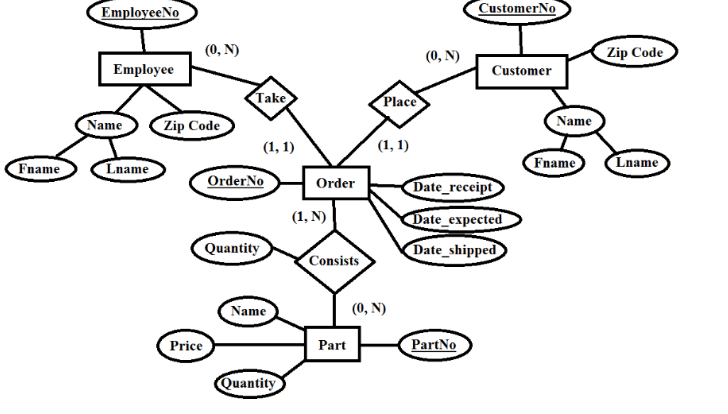


Q4. Consider a MAIL\_ORDER database in which employees take orders for parts from customers. The data requirements are summarized as follows: [20 points]

* The mail order company has employees, each identified by a unique employee number, first and last name, and Zip Code.
* Each customer of the company is identified by a unique customer number, first and last name, and Zip Code.
* Each part sold by the company is identified by a unique part number, a part name, price, and quantity in stock.
* Each order placed by a customer is taken by an employee and is given a unique order number. Each order contains specified quantities of one or more parts. Each order has a date of receipt as well as an expected ship date. The actual ship date is also recorded.

Design an entity–relationship diagram for the mail order database

Solution:



3+3+3+3+3+3+2(PK)

**Question 5 [ 45 Marks]**

Why SQL is called as structured query language ? **(1 Mark)**

Because it operates on data in structured form i.e. in the form of rows and columns.

What are different DML commands in SQL **(1 Mark)**

Select, Update, Delete, and Insert

Write names of 3 different join in SQL (**2 Marks)**

Inner, Outer, Cross, Equi, Natural

What is the difference between Nested Query and Corelated Nested Query ? **(2 Marks)** Give an example of each to show the difference **(2 Marks)** Which one is slower and why ? **(4 Marks)**

Difference and example can be found from our lecture slides.

Corelated Nested Query is **slower** because in Correlated Query,  Outer query executes first and for every Outer query row Inner query is executed. Hence, Inner query uses values from Outer query and therefore it is slow.

Using the below schema, write following SQL queries

*Person (driver-id, name, address )  
Car (license, model, year )  
Accident (report-number, date, location ) Owns (driver-id, license )*  *Participated (driver-id, license, report-number, damage-amount )*

1. Write SQL queries to create all of the above 5 tables with appropriate data types **(5 Marks)**

Create table Person (driver-id int, name varchar(20), address varchar(30));

Rest of the create queries are same

1. Write SQL queries to insert two rows each in “Person” and “Participated” tables **(2 Marks)**

Insert into Person values (10, ‘Ali’, ‘Islamabad’);

Rest of insert statements are same.

1. Write an SQL query to delete all those records of the “Person” table when *driver-id* is less than 10 **(2 Marks)**

**delete   
from Person where driver-id < 10;**

1. Write an SQL query to update the damage-amount in “Participated” by increasing 200 rupees for all those report-numbers which are greater than 1000 **(2 Marks)**

**Update Participated**

**Set damage-amount = damage-amount + 200**

**Where report-numbers > 1000;**

1. Write an SQL query to find total number of people who owned cars that were involved in accidents in 1989 **(2 Marks)**

**Select count(\*)**

**from Person P, Owns O**

**Where P.driver-id=O.driver-id**

**And P.driver-id IN (select driver-id From Participated)**

1. Write an SQL query that returns person name and license, If a person does not own any car, his name should still be displayed will an empty or null license **(2 Marks)**

**Select P.Name, O.License**

**from Person P, Owns O**

**Where P.driver-id=O.driver-id (+);**

A student can also write the other syntax of left outer join

1. Write an SQL query to delete the model Mazda belonging to “John Smith” **(2 Marks)**

**Delete from Car**

**Where model=’Mazda’**

**And license IN (select O.license**

**From Person P, Owns O**

**Where P.driver-id=O.driver-id**

**And P.name = ‘John Smith’);**

1. Write an SQL query that returns report number, date, license and damage-amount of a particular accident **(2 Marks)**

**Select P.*license, P.report-number,***

***P.damage-amount***

***from Participated P, Accident A***

**Where *P.*report-number = *A.*report-number**

**And A.Location = ‘Some city’;**

Using the below schema, write SQL queries.

*EmployeeInfo (EmpID, EmpFname, EmpLname, Department, Project, Address, DOB, Gender)*

*EmployeePosition( EmpID, EmpPosition, DateOfJoining, Salary)*

1. Write a query to fetch the number of employees working in the department ‘HR’ **(2 Marks)**

SELECT COUNT(\*) FROM EmployeeInfo WHERE Department = 'Cricket';

1. Write a query to find the first names of employees that begin with ‘S’ **(2 Marks)**

SELECT \* FROM EmployeeInfo WHERE EmpFname LIKE 'S%';

1. Write a query to fetch all the records from the EmployeeInfo table ordered by EmpLname in descending order and Department in the ascending order **(2 Marks)**

SELECT \* FROM EmployeeInfo ORDER BY EmpFname desc, Department asc;

1. Write a query to fetch details of employees whose EmpLname ends with an alphabet ‘A’ and contains five alphabets **(2 Marks)**

SELECT \* FROM EmployeeInfo WHERE EmpLname LIKE '\_\_\_\_\_a';

### Write a query to fetch all employees who also hold the managerial position (2 Marks)

SELECT E.EmpFname, E.EmpLname, P.EmpPosition

FROM EmployeeInfo E INNER JOIN EmployeePosition P ON

E.EmpID = P.EmpID AND P.EmpPosition IN ('Manager');

1. Write a query to fetch the department-wise count of employees sorted by department’s count in ascending order **(2 Marks)**

SELECT Department, count(EmpID) AS EmpDeptCount FROM EmployeeInfo GROUP BY Department ORDER BY EmpDeptCount ASC;

1. Write a query to retrieve duplicate records from a table **(2 Marks)**

SELECT EmpID, EmpFname, Department, COUNT(\*) FROM EmployeeInfo GROUP BY EmpID, EmpFname, Department HAVING COUNT(\*) > 1;

**\*\*\*\*\* The End \*\*\*\*\***