CSE0317:Database Management System

ASSIGNMENT-1

INSTRUCTIONS:

a. Neat presentation.
b. Include well labelled diagrams.
c. Write down differences in tabular format.
d. Mention and Highlight important text.
e. Use only blue and black ink for presentation

-:QUESTION/ANSWERS:-

1) Describe the characteristics of Database System. Write five applications of DBMS.

Ans

- A Database System is a combination of a Database and a Database Management System (DBMS). It is an organized collection of structured data, typically stored electronically in a computer system. The DBMS is the software that interacts with the database, users, and applications, providing a systematic way to create, retrieve, update and manage data.

- The characteristics of Database System are :

i. Data abstraction : Simplifies interaction with data by providing multiple levels of abstraction, allowing users to interact with data without needing to understand complex storage mechanisms.

ii. Data Integrity : Integrity constraints ensures that the data in the database remains accurate and consistent.

iii. Security : Security features control who can access or manipulate the data, protecting it from unauthorized access.

iv). Data Management : The system provides optimized storage and retrieval mechanisms, ensuring quick access to large amounts of data through indexing, query optimization, and other performance enhancing techniques.

- The applications of Database System are :

i). Banking and Finance : Used for managing customer accounts, processing transactions, handling loans, and other financial operations. A DBMS ensure the secure and efficient processing of banking data.

ii). Airline Reservations : Manages flight schedules, reservations, ticketing and customer details. It allows real-time processing, ensuring up-to-date information on seat availability and booking.

iii). Telecommunication : Handles customer data, billing, call records, and network management. It supports large-scale data processing and real-time data management for customer services.

iv). E-commerce : Manages product inventories, user data, orders, payment processing, and customer service. Its supports high-availability systems, secure transactions, and data analysis.

v). Healthcare : Stores and manages patient records, appointment scheduling, diagnostic information, and treatment histories. It ensures data is accessible to authorized personnel while maintaining privacy and data Integrity.

2). Explain Entity relationship model and list the various symbols used.

Ans

• The Entity-Relationship (ER) model is a conceptual framework used to describe the data structure of a database in terms of entities, attributes, and relationships. It provides a way to visually represent the logical structure of the database, making it easier to design and understand how data is organized and related within the system.

• The key components of the ER model are :

i). Entities : An Entity is a real-world object or concept that can have data stored about it. It represent things like a person, place, event or object.

ii). Attributes : Attributes are properties or characteristics of an entity. They describe the entity's details.

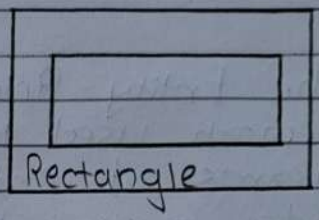
iii). Relationships : Relationships describe how entities are related to one another. A relationship connects two or more entities.

iv). Cardinality : Cardinality defines the number of instances of one entity that can be associated with instances of another entity in a relationship.

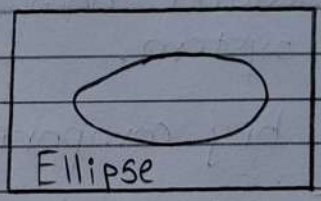
v). Primary Key : A Primary key is a unique attribute or a set of attributes that uniquely identifies each instance of an entity.

• The Symbols used in ER Model are :

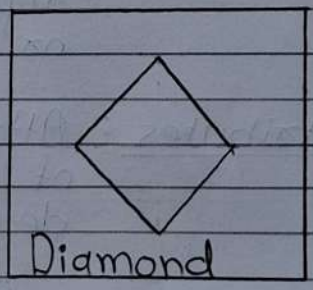
i). Rectangle (Entity) : Represents an entity. Each rectangle is labeled with the name of entity.



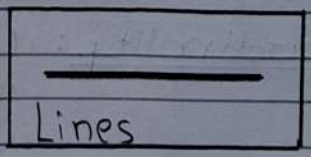
ii). Ellipse : Ellipse represent attributes in the ER Model.



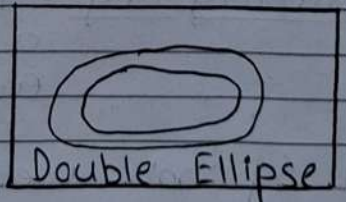
iii). Diamond : Diamonds represent relationships among entities.



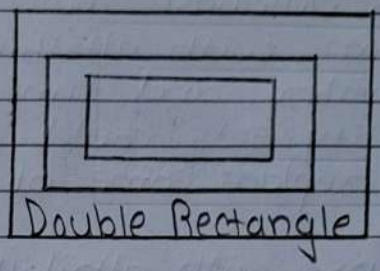
iv). Lines : Lines represent attributes to entities and entity sets with other relationship types.



v) Double Ellipse : Double Ellipses represent Multi-valued attributes.



vii. Double Rectangle: Double rectangle represents a weak entity.



3). Consider a University database for the scheduling of classrooms for final exams.
This database could be modelled as the single entity set exam, with attributes course-name, section-number, room-number, and time. Alternatively, one or more additional entity sets could be defined, along with relationship sets to replace some of the attributes of the exam entity set, as

- Course with attributes name, department and c-number.
- Section with attributes s-number and enrollement, and dependent as a weak entity set on course.
- Room with attributes r-number, capacity, and building.
- Exam with course-name, section-number, room-number, and time.

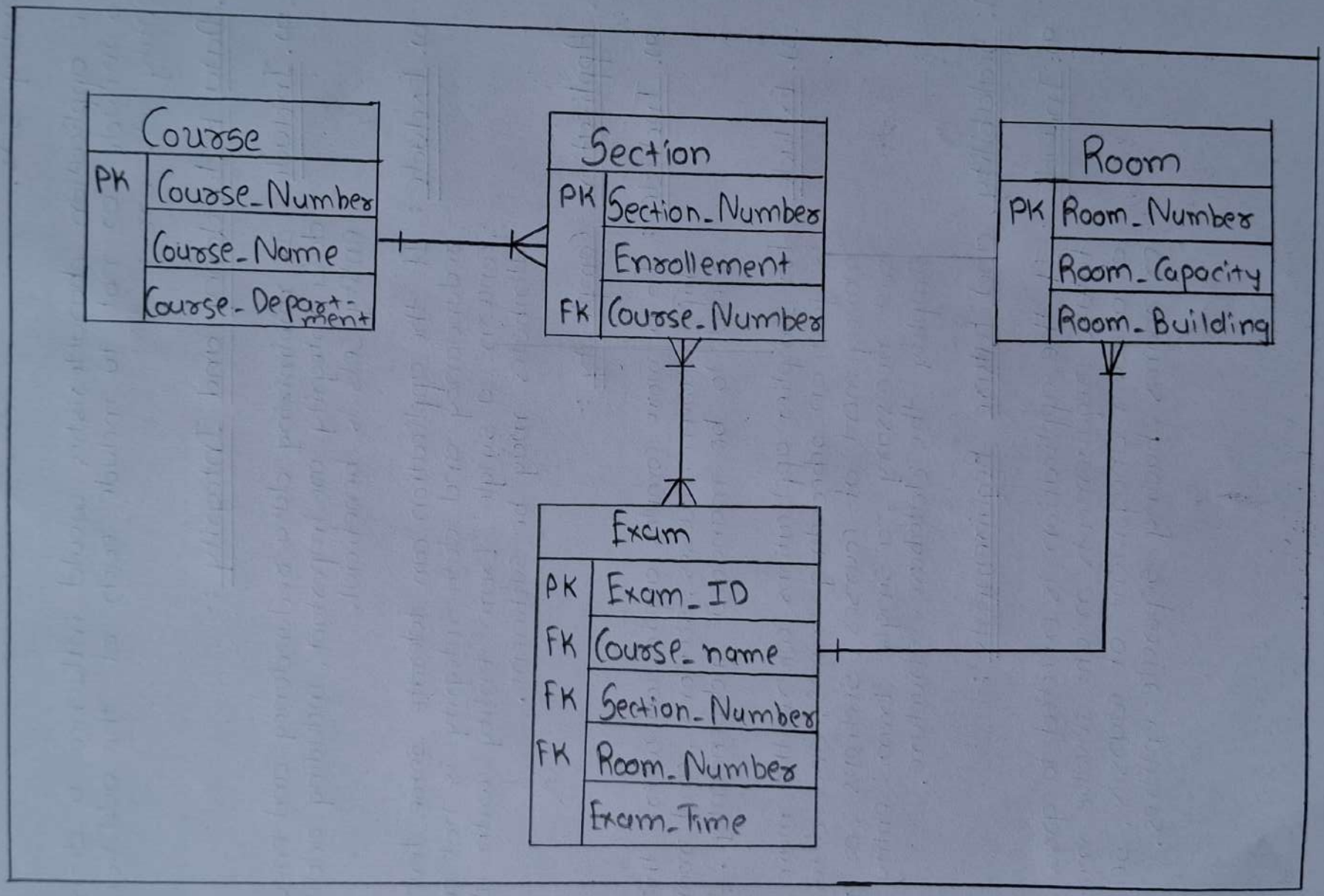
a). Show an ER-diagram illustrating the use of all three aditional entity sets listed.
b). Explain what application characteristics could influence a decision to include, or not to include each of the additional entity sets.

Ans

- The entities in these database are :

i). Exam : (course-name, section-number, Room-number, time)
ii). Course : (C_number, name, department)
iii). Section : (S_number, Enrollment, Course_name)
iv). Room : (R_number, Capacity, Building)

a). Show an ER-diagram illustrating the use of all three additional entity sets listed.


Course
PK
Course_Number
Course_Name
Course_Department
Section
PK
Section_Number
Enrollement
FK
Course_Number
Room
PK
Room_Number
Room_Capacity
Room_Building
Exam
PK
Exam_ID
FK
Course_name
FK
Section_Number
FK
Room_Number
Exam_Time

b). Explain what application characteristics would influence a decision to include or not to include each of the additional entity sets.

- The application characteristics would influence a decision to include or not to include each of the additional entity sets are :

i). Data Redundancy and Integrity :-

a). Include : If minimizing data redundancy and ensuring data integrity are important, including additional entity sets is beneficial.

b). Exclude : If the application can tolerate some level of redundancy and data integrity is not a primary concern, a single 'Exam' entity with all attributes may be sufficient.

ii). Application Complexity :-

a). Include : In a more complex application where the relationship between courses, sections, and rooms need to be managed independently.

b). Exclude : For simpler applications where the main focus is just on scheduling exams, and detailed management of courses, sections, or rooms isn't necessary, a single 'Exam' entity can simplify the database structure.

iii). Scalability and Future Requirements :-

a). Include : If the application is expected to grow or if future requirements might involve more detailed management of courses, sections, or rooms, having seperate entities.

b). Exclude : If the application has a fixed scope with no anticipated growth in complexity or data management requirements, a simple structure with a single 'Exam' entity may be more appropriate.

iv). Query Performance :-

a). Include : If the application needs to frequently perform complex queries that involve relationships between different entities, having seperate entities can optimize query performance and make the system more efficient.

b). Exclude : If the queries are simple and mostly focused on the exam schedule, a single 'Exam' entity might suffice and could reduce the overhead of joining multiple tables.

v). Maintainability and Ease of updates :-

a). Include : In application where maintainability is a concern, seperating entities can makes updates easier and less error-prone.

b). Exclude : If the application is small-scale and maintainability is not a significant concern, a simpler structure with a single entity might be easier to implement and manage in the short term.

vi). Data Independence :-

a). Include : If the application needs to support data independence - where changes to the schema at one level should not affect other parts of the application then seperate entity sets are preferable.

b). Exclude : In cases where data independence is not a requirement and the application design is static, using a single entity set might be adequate.

4). Define the term Aggregation. Explain with Example.

Ans

- Aggregation is a concept in the Entity-Relationship (ER) model where a relationship between entities is treated as a higher-level entity. This is particularly useful when you want to model a scenario where a relation itself participates in another relationship. It is used to express a relationship between relationships, which allows for a more complex and nuanced representation of real-world scenarios.

- Let's understand Aggregation with an example :

Consider a scenario where a university database is being designed to schedule project reviews by a committee of professors. The scenario involves the following :

i). Entities :-

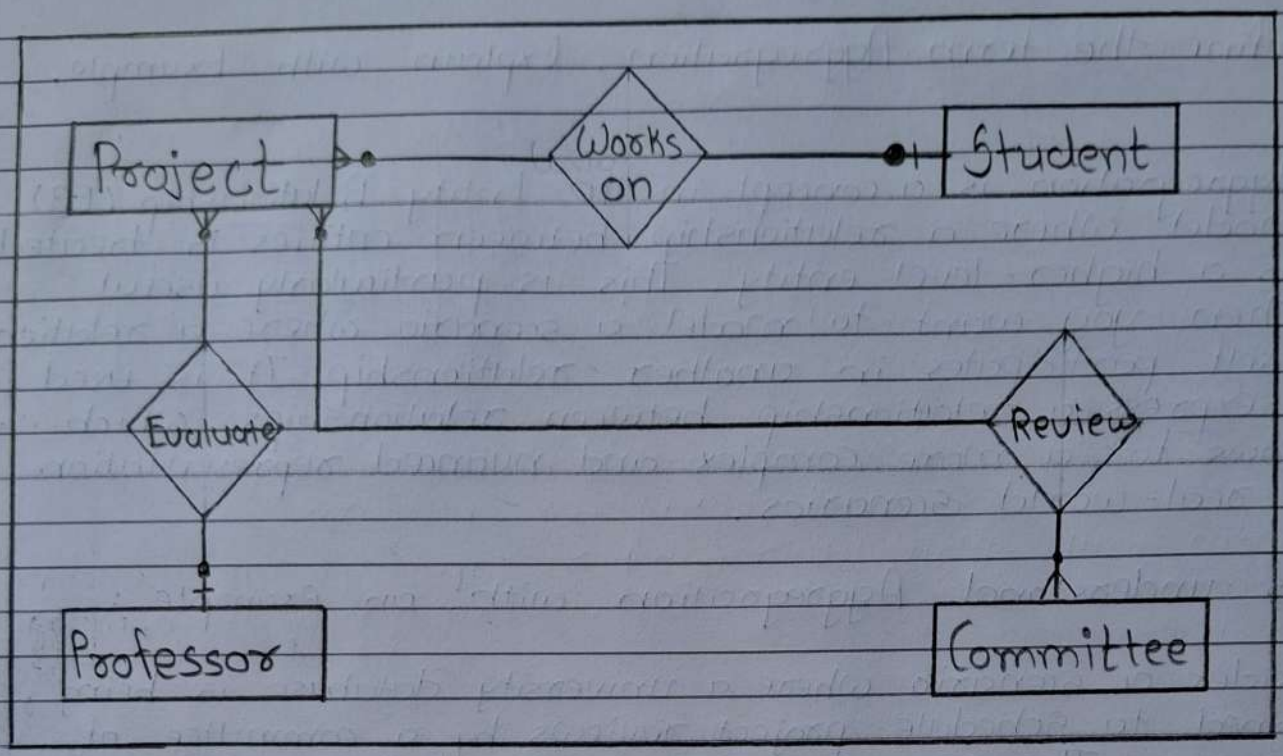
- 'Project' : Attributes include 'Project_ID', 'Title'.
- 'Student' : Attributes include 'Student_ID', 'Name'.
- 'Professor' : Attributes include 'Professor_ID', 'Name'.
- 'Committee' : Attributes include 'Committee_ID', 'Formation_Date'.

ii). Relationships :-

- 'Works_On' : A Relationship between 'Student' and 'Project'.
- 'Evaluates' : A relationship between 'Project' and 'Professor'.
- 'Review' : A relationship between 'Committee' and 'Project'.

- So, Here we have two-three relationship between 'Student', 'Professor', 'Committee' with 'Project' entity.

- Let's understand it with diagram representation :


Project
Works on
Student
Evaluate
Review
Professor
Committee

5). Describe how an ER Schema can be converted to tables. Also, explain how any weak entity set is converted to a strong entity set.

Ans

- Converting an Entity-Relationship (ER) schema into a set of tables involves transforming entities, attributes, relationships, and any constraints into a relational database structure. Here's how each component of an ER Schema is typically converted into tables:

i). Entities
ii). Relationships
iii). Attributes
iv). Weak Entities

- If you want to convert a weak entity set to a strong entity set here's as follows:

i). Identify the weak Entity Attributes :-

- Take all the attributes of the weak entity, including its identifying relationship with the strong entity.

ii). Assign a New Primary Key :-

- Introduce a new unique attribute (like 'Dependent_ID' for 'Dependent') to act as the primary key.

iii). Eliminate the need for the Strong Entity's Key :-

- The new primary key makes the entity independent, eliminating the need to rely on the primary key of the strong entity.

iv). Adjust Relationships :-

- If the weak entity had any identifying relationship, modify it to reflect that it is now a strong entity.

6). What is an attribute? Explain the following attribute types:
a. Simple and composite
b. Single-valued and multivalued
c. Null attributes
d. Derived attributes

Ans

- An attribute is a property or characteristics of any entity in a database. In the context of an Entity Relationship (ER) model, an attribute represents a piece of data associated with an entity or relationship.

- For Example: In an entity 'Student', attributes could include 'Student_ID', 'Name', 'Date_of_Birth'. Each attribute has a specific domain, which defines the type of values it can take.

- The types of attributes are:

i). Simple and Composite
ii). Single-valued and multivalued
iii). Null attributes
iv). Derived attributes

i). Simple and Composite Attributes :-

a). Simple Attributes :-

- A simple attributes is an attribute that cannot be divided further. It is atomic and contains a single value. like 'Student_ID', 'Age' etc.

b). Composite Attributes :-

- A composite attribute is one that can be divided into smaller subparts, each of which represents a more

- basic attribute with its own meaning. like the 'name' attribute can be composite, consisting of 'First_name', 'Last_name'.

ii). Single-Valued and Multivalued Attributes :-

a). Single-valued attributes :-

- A Single-valued attributes is an attribute that holds a single value for a given entity like 'Age', 'Student_ID'.

b). Multi-valued attributes :-

- A multi-valued is one that can hold multiple values for a single entity like 'Phone_number', 'Email_address'.

iii). Null Attributes :-

a). Null Attributes :-

- A Null attributes represents an attribute that does not have a value in a specific entity instance. This could be because the value is unknown, not applicable, or not yet assigned. A 'Middle_name' attribute might be null for some students who don't have a middle name. Similarly, a 'Date_of_Graduation' might be null for students who haven't yet graduated.

iv). Derived Attributes :-

a). Derived Attributes :-

- A derived attributes is an attribute whose value is calculated or derived from other

attributes in the database. It is not stored directly in the database but can be computed when needed. An 'Age' can be derived from the 'Date_of_Birth' attribute by subtracting the birth date from the current date. Similarly, 'Total_invoice' in an invoice can be derived from 'Quantity' and 'Unit_Price'.

7). Describe the term relationships in Relational Data Modeling. How are they associated with Cardinality Ratios? Explain with an example.

Ans

- In relational data modeling, a relationship is an association or link between two or more entities in a database. Relationships describe how entities interact with each other and define the rules for these interactions. They are crucial for structuring the database and ensuring that related data is connected logically.

- For example, in a university database :

The Student entity might be related to the Course entity through a 'Registers' relationship, indicating that a Student can register for courses.

- The cardinality ratio defines the rules of how the entities participate in the relationship are :

i). Determining the Type of Join :-

- In a database schema, the type of cardinality affects how tables are joined. Like, in a One-to-Many relationships typically result in a foreign key being placed in the "many" table to refer back to the 'one' table.

ii). Enforcing Data Integrity :-

- Cardinality ratioes helps ensure that the database reflects real-world constraints. Like, in a One-to-One relationship might be enforced with unique constraints on the foreign key.

iii). Optimizing Queries :-

- Knowing the cardinality can help in optimizing queries. For instance, a Many-to-One relationship can be optimized by indexing the foreign key in the "many" table.

- Let's understand Relational Data modeling with an example:

→ Suppose we have a 'Student' and a 'Course' table with the following relationship:

i). Entities : 'Student' (Student_ID, Name) and 'Course' (Course_ID, Title)

ii). Relationship : "Enrolls"

iii). Cardinality : Many-to-Many

→ To represent this relationship in a relational database, we create a junction table 'Enrollement':

```
CREATE TABLE Enrollment (
        Student_ID INT,
        Course_ID INT,
        Enrollment_Date DATE,
        PRIMARY KEY (Student_ID, Course_ID),
        FOREIGN KEY (Student_ID) REFERENCES
                                Student (Student_ID),
        FOREIGN KEY (Course_ID) REFERENCES
                                Course (Course_ID)
    );
```

8). Consider the following relational Schema :

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
account (account_number, branch_name, balance)
depositor (customer_name, account_number)
borrower (customer_name, loan-number)
loan (loan_number, branch_name, amount)

Answer the following queries in relational algebra :

a). Find the loan number for each loan of an amount greater than $1200.

b). Find the name of all customers of the Bank who have a loan, an account, or both from the bank.

c). Find the customers of the bank who have an account but not a loan.

d). Find the names of all customers with loans at the perryridge branch.

e). Find all customers who have an account at all branches located in Brooklyn city.

Ans

a). Find the loan number for each loan of an amount greater than $1200.

$$\Pi_{\text{loan\_number}}(\sigma_{\text{amount} > 1200}(\text{loan}))$$

b). Find the name of all customers of the Bank who have a loan, an account, or both from the bank

$$\Pi_{\text{customer\_name}}(\text{depositor}) \cup \Pi_{\text{customer\_name}}(\text{borrower})$$

c). Find the customers of the bank who have an account but not a loan.

$$\Pi_{\text{customer\_name}}(\text{depositor}) - \Pi_{\text{customer\_name}}(\text{borrower})$$

d). Find the names of all customers with loans at the perryridge branch.

$$\Pi_{customer\_name}(\sigma_{branch\_name = 'Perryridge'}(loan) \bowtie borrower)$$

e). Find all customers who have an account at all branches located in Brooklyn city.

$$\Pi_{customer\_name}((depositor \bowtie account) \div \Pi_{branch\_name}(\sigma_{branch\_city = 'Brooklyn'}(branch)))$$

9). Define constraints. Explain referential integrity constraints. How can it be implemented using SQL?

Ans

- Constraints are rules or restrictions applied to the data in a database to ensure data integrity, accuracy and consistency. These constraints are enforced by the database management system (DBMS) when data is inserted, updated, or deleted. If any operation violates a constraint, the DBMS will reject that operation.

- The types of constraints are :

i). Primary key Constraint
ii). Unique Constraint
iii). Not Null Constraint
iv). Check Constraint
v). Foreign key Constraint

- Referential Integrity is a type of constraints that ensure the consistency of relationships between tables in a relational database. It guarantees that a foreign key in a table matches a primary key in another table, thereby linking the two tables correctly. This prevents "orphaned" records and ensures that relationships between tables are preserved.

- Referential Integrity works as the below :

i). Parent Table : The table containing the primary key.

ii). Child Table : The table containing the foreign key.

The foreign key in the child table must correspond to an existing primary key value in the parent table.

- Let us understand the implementation of Referential integrity in SQL :

Step-1 : Creating the Parent Table (Customer)

```
CREATE TABLE Customer (
        Customer_ID INT PRIMARY KEY,
        Name VARCHAR(100),
        Address VARCHAR(200)
);
```

Step-2 : Creating the Child Table (Order) with a Foreign Key.

```
CREATE TABLE Order (
        ORDER_ID INT PRIMARY KEY,
        ORDER_Date DATE,
        Customer_ID INT,
        FOREIGN KEY (Customer_ID) REFERENCES
                        Customer (Customer_ID)
);
```

∴ The 'Customer' table has 'Customer_ID' as Primary key and the 'Order' table has 'Customer_ID' as a foreign key that references 'Customer_ID' in the 'Customer' table.

- When you define a foreign key constraint in the 'Order' table, the database enforces the following rules :

i). We cannot insert a record into the 'Order' table with a 'Customer_ID' that does not exist in the 'Customer' table.

ii). You cannot delete a record from the 'Customer' table if that 'Customer_ID' is referenced by records in the 'Order' table, unless you first delete or update those referencing records.

iii). You cannot update a 'Customer_ID' in the 'Customer' table to a new value unless we also update or remove corresponding 'Customer_ID' values in the 'Order' table.

- You can specify what happens when a referenced primary key in the parent table is updated or deleted using options like 'ON DELETE' or 'ON UPDATE'. The options include :

i). CASCADE : Automatically delete or update the matching records in the child table when a record in the parent table is deleted or updated.

ii). SET NULL : Set the foreign key in the child table to 'NULL' when referenced record in the parent child is deleted or updated.

iii). No Action : Prevent deletion or update if there are matching records in the child table.

```
CREATE TABLE Order (
        Order_ID INT PRIMARY KEY,
        Order_Date DATE,
        Customer_ID INT,
        FOREIGN KEY (Customer_ID) REFERENCES
                                    Customer ( Customer_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

- In this example,

i). If a 'Customer_ID' is deleted from the 'Customer' table, all related orders in the 'Order' table will also be deleted ('ON DELETE CASCADE').

ii. If a 'Customer_ID' is updated in the 'Customer' table, the change will automatically propogate to the 'Order' table ('ON UPDATE CASCADE').

10). Differentiate between

a). File system and DBMS.
b). Instance and Schema
c). Generalization and specialization

Ans

a). File System and DBMS :-

	Aspect	File System	DBMS
1.	Data Redundancy	High data redundancy. Data is often duplicated in multiple files	Minimize data redundancy through normalization and controlled redundancy.
2.	Data Integrity	Difficult to enforce data integrity. No centralized control.	Enforce data integrity using constraints and rules.
3.	Data Access	Limited, often requires writing complex program to access data.	Provides high-level query languages.
4.	Data Security	Basic file-level security, no finegrained access control	Fine-grained access control, role-based security, and encryption.
5.	Data Backup and Recovery	Manual and often complex process, prone to errors.	Automated backup and recovery processes with robust tools.

6.	Concurrency Control	Difficult to manage concurrent access, leading to issues like data inconsistency.	Supports multi-user environments with concurrency control mechanisms.
7.	Data Relationships	Relationships are managed by the application, leading to complex code.	Relationships are managed within the database using foreign keys, joins etc.
8.	Data Abstraction	Low users must understand the physical data structure	High supports different levels of data abstraction (Physical, logical, view).
9.	Data Backup and Recovery	Manual and often complex process, prone to errors.	Automated backup and recovery processes with robust tools

b). Instance and Schema :-

Sr	Aspect	Instance	Schema
1.	Definition	The actual data stored in database at a particular moment in time.	The overall design or structure of the database, including tables, relationships, and constraints
2.	Level	Corresponds to a snap-shot of the data	Corresponds to logical structure of the data

3.	Dynamic vs Static	Dynamic changes frequently as data is added, updated, or deleted.	Static defines the structure and changes infrequently.
4.	Focus	Focuses on the state of the data at a specific point in time.	Focuses on how the data is organized and how it relates to other data in the database.
5.	Example	A specific set of data in a 'Student' table at a given time.	The blueprint or layout of the 'Student' table

C). Generalization and Specialization :-

Sr.	Aspect	Generalization	Specialization
1.	Definition	The process of combining multiple entities that share common features into a single, more general entity.	The process of creating new entities by distinguishing features of an existing general entity.
2.	Bottom-Up Vs Top-Down	Bottom-Up approach, starts with specific entities and abstracts them into a more general form.	Top-down approach, starts with a general entity and creates more specialized sub-entities.

3.	Purpose	To reduce redundancy by grouping similar entities and attributes into a common entity.	To differentiate between entities based on specific characteristics and create more precise models.
4.	Focus	Focuses on finding commonalities among entities.	Focuses on identifying unique characteristics to create specialized entities.
5.	Output	A higher-level, generalized entity that encompasses the features of the specific entities.	Lower-level, specialized entities that inherit features from the general entity but also have distinct attributes
6.	Example	Combining 'Car', 'Truck', 'Motorcycle' into a generalized 'Vehicle'	Dividing a 'Vehicle' entity into more specialized entities like 'Car', 'Truck' and 'Motorcycle'