Q.NO.1

a. Discuss any THREE (3) current threats of modem databases.

Ans:-

1. Data Breaches:- Data breaches are a significant threat to modern databases. Attackers may exploit vulnerabilities in database systems to gain unauthorized access to sensitive information. This could result in the theft of personal data, financial information, or intellectual property.

2. SQL Injection Attacks:- SQL injection attacks occur when malicious SQL statements are inserted into user input fields, tricking the database into executing unintended commands. This can lead to data manipulation, unauthorized access, and even data deletion.

3. Ransomware:- Ransomware attacks target databases by encrypting the data and demanding a ransom for decryption keys. If not paid, organizations risk losing access to their critical data.

b. Explain data redundancy with an example.

Ans:- Data redundancy refers to the duplication of data within a database system. This redundancy can lead to data inconsistencies and increased storage requirements.

Example: Consider a company's employee database where each employee's address is stored along with their personal information. If two employees, John and Jane, share the same address, storing the address information twice creates data redundancy. If the address needs to be updated, it must be changed in multiple places, increasing the chances of inconsistencies.

c. Describe any FOUR (4) functions of a database administrator.

Ans:-

1. Data Security and Authorization:- Database administrators (DBAs) manage user access and permissions to ensure that only authorized users can view, modify, or delete data. They also implement security measures to protect the database from external threats.

2. Database Performance Tuning:- DBAs monitor database performance and optimize it for efficiency. This includes optimizing queries, indexing, and configuring hardware resources to ensure the database operates smoothly.

3. Backup and Recovery:- DBAs are responsible for implementing backup and recovery strategies to safeguard data against loss or corruption. They regularly schedule backups and ensure that data can be restored in case of a disaster.

4. Database Design and Schema Management:- DBAs play a role in designing the database schema and making schema modifications when necessary. They ensure that the database structure aligns with the organization's requirements and evolves as needed.

Q.NO.2

a. Describe the activities performed during the following phases of database design and development process:

i. Requirement Analysis

ii. Conceptual Database Design

iii. Logical Database Design

iv. Physical Database Design

Ans:-

i. Requirement Analysis:- In this phase, database designers gather and analyze the requirements of the system. This involves interviewing stakeholders, identifying data entities, defining relationships, and determining data constraints.

ii. Conceptual Database Design:- During this phase, designers create a high-level conceptual model of the database. This includes defining entities, attributes, and relationships without considering specific implementation details. Entity-relationship diagrams (ERDs) are often used.

iii. Logical Database Design:- In this phase, designers translate the conceptual model into a logical model that can be implemented in a database management system (DBMS). This includes defining tables, keys, constraints, and normalizing the data structure.

iv. Physical Database Design:- This phase involves designing the physical storage structures for the database, including specifying file organization, indexing strategies, and hardware considerations. It aims to optimize performance and storage efficiency.

b. Define the attributes listed below with an example:

i. Derived attribute

ii. Multivalued attribute

iii. Composite attribute

Ans:-

i. Derived Attribute:- A derived attribute is one whose value is derived from other attributes in the database. It is not directly stored but can be computed from other attributes.

Example: In a database for tracking employee records, the "Age" attribute can be a derived attribute calculated from the "Date of Birth" attribute. The database system calculates the age of each employee based on their birthdate.

ii. Multivalued Attribute:- A multivalued attribute is an attribute that can have multiple values for a single entity. It is typically represented as a set or a list of values.

Example: In a database for a library, the "Authors" attribute of a book entity can be multivalued because a book can have more than one author. The "Authors" attribute would contain a list of author names.

iii. Composite Attribute:- A composite attribute is an attribute that can be divided into smaller, meaningful sub-parts, which represent more basic attributes with independent meanings.

Example: In a database for tracking addresses, the "Address" attribute can be a composite attribute consisting of sub-attributes like "Street Address," "City," "State," and "ZIP Code." Each sub-attribute has its own meaning and can be manipulated individually.

c. Draw an example of entity relationship of the followings:

i. Unary relationship

ii. Binary relationship

iii. Ternary relationship

Ans:-

i. Unary Relationship:- A unary relationship occurs when an entity is related to itself.

Example: In a database for a social networking platform, a "Friendship" unary relationship can exist between the "User" entity and itself. Each "User" can be connected to other "Users" through this relationship to represent friendships.

ii. Binary Relationship:- A binary relationship involves two different entities and describes how they are related to each other.

Example: In a university database, a "Teaches" binary relationship can exist between the "Professor" entity and the "Course" entity. It shows which professor teaches which course.

iii. Ternary Relationship:- A ternary relationship involves three different entities and describes how they are related to each other.

Example: In a hospital database, a "Patient-Doctor-Room" ternary relationship can exist to show which doctor is assigned to which patient in which room. This relationship captures the allocation of doctors, patients, and rooms in the hospital.

Q.NO.3

a. Describe the following terms:

i. Candidate key

ii. Primary Key

iii. Foreign Key

iv. Tuple

Ans:-

i. Candidate Key:- A candidate key is a set of one or more attributes (columns) in a relational database table that can uniquely identify each tuple (row) in that table. It means that no two tuples can have the same combination of values in the candidate key attributes. A table can have multiple candidate keys, but one of them is chosen as the primary key.

ii. Primary Key:- A primary key is a specific candidate key chosen to uniquely identify tuples in a relational database table. It must have the following properties:

- Unique: No two tuples can have the same primary key value.
- Not Null: The primary key attribute cannot have a NULL value.
- Stable: The values in the primary key should not change over time.

iii. Foreign Key:- A foreign key is an attribute or set of attributes in one table that refers to the primary key of another table. It establishes a relationship between the tables, ensuring referential integrity. Foreign keys are used to maintain data consistency and enforce relationships between tables.

iv. Tuple:- In the context of a relational database, a tuple is a single row or record in a table. It represents a complete set of attribute values for an entity in the database. Each tuple contains data for all the columns defined in the table schema.

b. Distinguish between cardinality and degree of a relation.

Ans:-

Cardinality: Cardinality refers to the number of tuples (rows) in a relation (table). It describes the "how many" aspect of a relationship. For example, if you have a "Students" table, the cardinality of this table might be the number of students enrolled, indicating how many students are in the database.

Degree: Degree, on the other hand, refers to the number of attributes (columns) in a relation (table). It describes the "how much" aspect of a relation. For example, if you have a "Students" table with attributes like student ID, name, and age, the degree of the table is 3, indicating there are three attributes.

c. Suppose you are given the following requirements for a simple database for the Nepal National Hockey League (NNHL):

• the NNHL has many teams,

• each team has a name, a city, a coach, a captain, and a set of players,

• each player belongs to only one team,

• each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records, • a team captain is also a player,

• a game is played between two teams (referred to as host team and guest team) and has a date (such as May 11th, 2022) and a score (such as 4 to 2).

Construct a clean and concise ER diagram for the NNHL database using the Crow Foot notation.

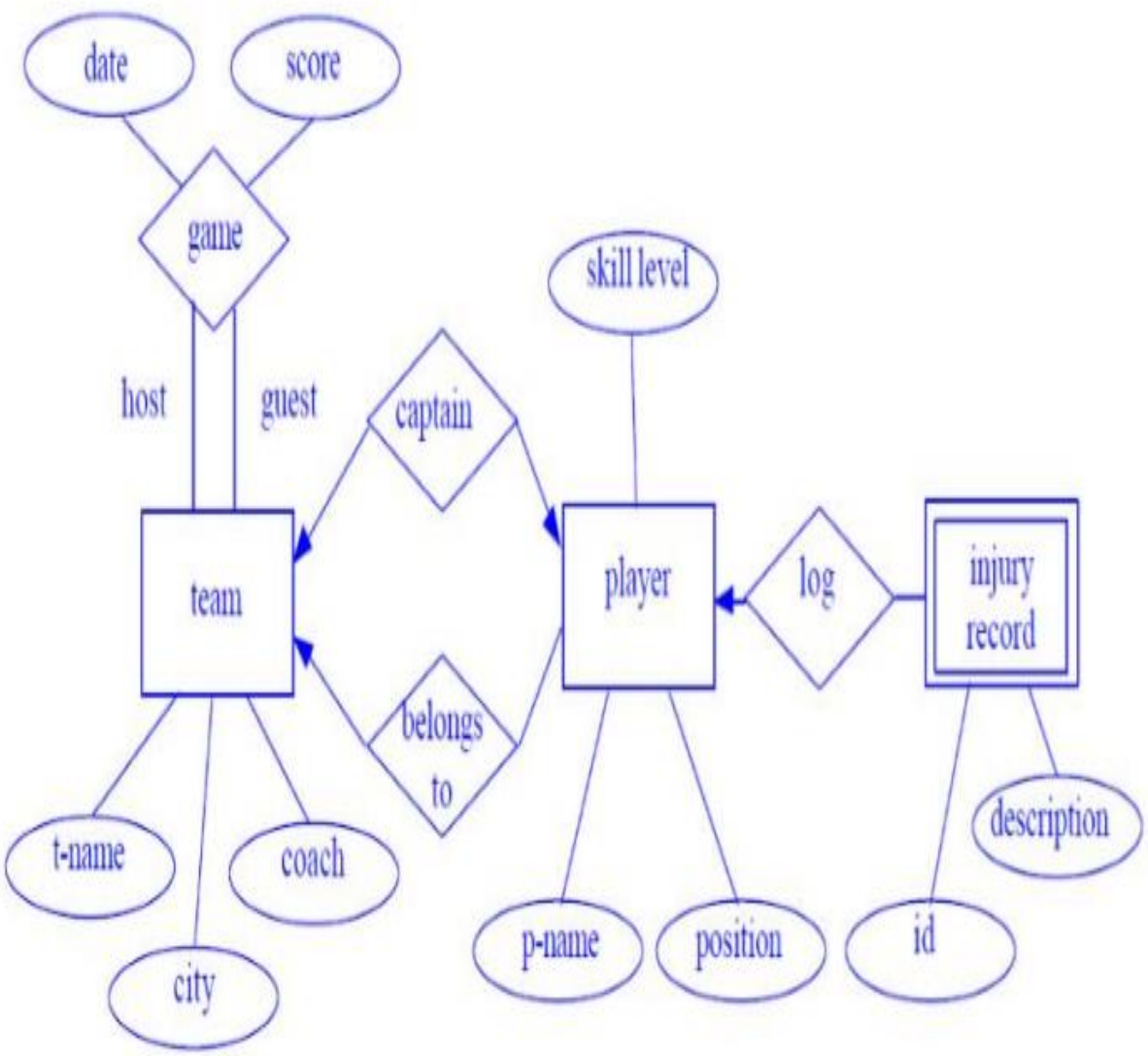
Ans:-

Entities:

- Team (Attributes: TeamID [Primary Key], Name, City, Coach)
- Player (Attributes: PlayerID [Primary Key], Name, Position, Skill Level)
- InjuryRecord (Attributes: RecordID [Primary Key], PlayerID [Foreign Key], InjuryDetails)
- Game (Attributes: GameID [Primary Key], Date, HostTeamID [Foreign Key], GuestTeamID [Foreign Key], HostTeamScore, GuestTeamScore)

Relationships:

- Team-Player Relationship: One-to-Many (One Team has Many Players)
- Player-InjuryRecord Relationship: One-to-Many (One Player has Many Injury Records)
- Game-Team Relationship (Two instances): Many-to-One (One Game has One Host Team and One Guest Team)
- Player-Captain Relationship: One-to-One (One Player can be the Captain of One Team)



Q.NO.4

a. Differentiate Functional Dependency and Transitive Dependency.

Ans:-

Functional Dependency:- In a relation, attribute Y is said to be functionally dependent on attribute X if, for every unique value of X, there is a unique value of Y. It means that the value of Y can be determined by the value of X. This forms the basis for defining keys and ensuring data integrity.

Transitive Dependency:- Transitive dependency occurs when an attribute Y depends on another attribute Z, which in turn depends on attribute X. In other words, Y is functionally dependent on X indirectly through Z. Transitive dependencies should be eliminated through normalization to ensure that data is stored efficiently and without redundancy.

b. Discuss the following database normalization levels:

i. 1 Normal form

ii. 2" Normal Form

iii. 3rd Normal Form

Ans:-

i. 1st Normal Form (1NF):- In 1NF, all attributes in a relation must have atomic (indivisible) values. Each column should contain only a single value, not a list or set of values. This eliminates repeating groups and ensures that data is organized into rows and columns.

ii. 2nd Normal Form (2NF):- To achieve 2NF, a relation must first be in 1NF, and then it should eliminate partial dependencies. This means that all non-key attributes should be functionally dependent on the entire primary key, not just part of it. This level of normalization prevents redundancy and ensures that each piece of data is stored in only one place.

iii. 3rd Normal Form (3NF):- 3NF builds upon 2NF and further eliminates transitive dependencies. In a 3NF relation, every non-key attribute should be functionally dependent only on the primary key, and there should be no indirect dependencies through other non-key attributes. This level of normalization reduces data redundancy and improves data integrity.

c.

Table 1-Student_Grade

Std_ID	Std_Name	Course_ID	Course_Name	Grade	Faculty_ID	Faculty_Name	Faculty_Phone
1	Kim	EC3119	Database	A	1178	Martin	06-8502338
1	Tila	EC3340	Information System	A	1167	Andrew	06-8502334
2	Ying Ying	EC3119	Database	A	1178	Martin	06-8502338
3	Teng Hui	EC3119	Database	A	1178	Martin	06-8502338

Normalize the above table until 3NF.

Ans:-

Table 1: Student

Std_ID	Std_Name
1	Kim
2	Ying Ying
3	Teng Hui

Table 2: Courses

Course_ID	Course_Name
EC3119	Database
EC3340	Information System

Table 3: Student_Grades

Std_ID	Course_ID	Grade	Faculty_ID	Faculty_Name	Faculty_Phone
1	EC3119	A	1178	Martin	06-8502338
1	EC3340	A	1167	Andrew	06-8502334
2	EC3119	A	1178	Martin	06-8502338
3	EC3119	A	1178	Martin	06-8502338

These tables are now in 3NF, and data redundancy has been minimized.

Q.NO.5

a. Define the function of each clauses listed below:

i. WHERE

ii. GROUP BY

iii. HAVING

iv. FULL OUTER JOIN

v. NATURAL JOIN

Ans:-

i. WHERE:- The WHERE clause is used in a SQL query to filter rows from a table based on a specified condition. It allows you to retrieve only those rows that meet the specified criteria. For example, you can use the WHERE clause to select rows where a certain column's value is equal to, greater than, less than, or not equal to a specific value.

ii. GROUP BY:- The GROUP BY clause is used in conjunction with aggregate functions (such as SUM, COUNT, AVG, etc.) to group rows from a table into sets based on the values in one or more columns. It is often used in combination with SELECT statements to perform operations on each group of rows separately. For instance, you can use GROUP BY to calculate the total sales for each product category.

iii. HAVING:- The HAVING clause is used in combination with the GROUP BY clause to filter the results of a grouped query based on aggregate function results. It allows you to specify conditions that must be met by groups of rows, effectively filtering out groups that don't satisfy the specified conditions. For example, you can use HAVING to select only product categories with total sales greater than a certain threshold.

iv. FULL OUTER JOIN:- A FULL OUTER JOIN is a type of SQL join that combines rows from two or more tables based on a specified condition and includes all matching and non-matching rows from both tables. In the result set, you'll get all the rows from the left table and all the rows from the right table, with NULL values filling in for non-matching rows on either side. This join type ensures that no data is lost from either table.

v. NATURAL JOIN:- A NATURAL JOIN is a type of SQL join that combines rows from two or more tables based on columns with the same name and data type in both tables. It essentially performs an implicit join by matching columns with the same name in both tables without the need to specify the join condition explicitly. However, it can be risky because if the column names change or if there are additional columns with the same name, it can lead to unexpected results.

b.

Table 2 - Customer

CustomerlD	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Marla Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. De la constitucion	Mexico D.F.	05021	Mexico
3	Antonao Moreno Taqueria	Antonio moreno	Mataderos 2312	Mexico D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq	London	Wa1 1DP	UK
5	Berglunds snabbkop	Christina Berglund	Berguvsagen B	Lulea	S - 958 22	Sweden

Write SQL Statement for the following queries referring to table 2:

i. Select all customers with a CustomerName that have "r" in the second position

ii. Select all customers from the "Customers" table, sorted ascending by the 'Country' and descending by the "CustomerName" column

iii. Insert a new record in the "Customers" table.

iv. List the number of customers in each country:

v. Update the first customer (CustomerlD = 1) with a new contact person and a new city.

vi. List the number of customers in each country. Only include countries with more than 5 customers.

Ans:-

```
//i. Select all customers with a CustomerName that have "r" in the second position:
SELECT * FROM Customer
WHERE SUBSTRING(CustomerName, 2, 1) = 'r';

//ii. Select all customers from the "Customers" table, sorted ascending by the 'Country'
and descending by the "CustomerName" column:
SELECT * FROM Customer
ORDER BY Country ASC, CustomerName DESC;

//iii. Insert a new record in the "Customers" table:
INSERT INTO Customer (CustomerID, CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES (6, 'New Customer', 'John Doe', '123 Main St', 'Anytown', '12345', 'USA');


//iv. List the number of customers in each country:
SELECT Country, COUNT(*) AS CustomerCount
FROM Customer
GROUP BY Country;

//v. Update the first customer (CustomerID = 1) with a new contact person and a new city:
UPDATE Customer
SET ContactName = 'New Contact Person', City = 'New City'
WHERE CustomerID = 1;

//vi. List the number of customers in each country. Only include countries with more than 5 customers:
SELECT Country, COUNT(*) AS CustomerCount
FROM Customer
GROUP BY Country
HAVING COUNT(*) > 5;
```