20CS403/20CB403/20DS403/20IT403

Hall Ticket Number:

II/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

July/August, 2023 **Common to CB, CSE,DS & IT Branches**

Fourth Semester **Database Management Systems**

Time: Three Hours **Maximum:** 70 Marks

Answer question 1 compulsory. **(14X1 = 14Marks)**

Answer one question from each unit. **(4X14=56 Marks)**

			CO	BL	M
1	a)	List out the applications of DBMS.	CLO-1	L1	1M
	b)	Define single valued and multivalued attributes.	CLO-2	L2	1M
	c)	What are the disadvantages of file processing system?	CLO-1	L1	1M
	d)	Give the levels of data abstraction.	CLO-2	L2	1M
	e)	Give the syntax and example for CREATE command in SQL.	CLO-3	L2	1M
	f)	Give an example for foreign key.	CLO-2	L2	1M
	g)	What is Natural Join? Give an example.	CLO-3	L1	1M
	h)	What is B+ tree?	CLO-5	L1	1M
	i)	What are the properties of decomposition?	CLO-4	L1	1M
	j)	What is Multi value dependency? Give an example	CLO-4	L1	1M
	k)	What is the use of checkpoints in recoverability?	CLO-5	L2	1M
	l)	When the transaction will be rolled back?	CLO-6	L2	1M
	m)	What is a Time stamp?	CLO-6	L1	1M
	n)	Define strict schedule.	CLO-5	L2	1M
		Unit-I			
2	a)	Describe Three schema architecture and Data independence in detail	CLO-1	L2	7M
	b)	Define DBMS. List Database users and their roles	CLO-1	L2	7M
		(OR)			
3	a)	Discuss in detail about the main steps in the database design and clearly focus in detail about the goal of each step? In which steps is the ER model mainly used?	CLO-2	L3	7M
	b)	Design an ER Model for College Admission System.	CLO-2	L3	7M
		Unit-II			
4	a)	Explain in detail about different types of attributes with diagrammatical representation in ER model.	CLO-4	L5	7M
	b)	What is aggregate function? Explain all aggregate functions with the help of example.	CLO-3	L2	7M
		(OR)			
5	a)	State various fundamental operations in the relational algebra and explain them with suitable examples.	CLO-3	L3	7M
	b)	Explain in detail about different types of joint operations with example.	CLO-2	L2	7M
		Unit-III			
6	a)	List and explain various functional dependency mechanism.	CLO-3	L2	7M
	b)	Given a relation with attributes R= (A,B,C,D,E,F,G,H) and following functional dependencies. CH→G A→BC B→CFG E→A F→EG State the strongest normal form that the relation R is in. If R is not in BCNF, decompose it into a collection of BCNF relations.	CLO-5	L5	7M

(OR)

7	a)	Construct B tree to insert the following key values (order of the tree is 4) 10,50,,5,7,2,345,55,22,44,56,58,70,17,6,21	CLO-3	L3	7M
	b)	What is meant by normalization? Is it mandatory in a relational database design? Briefly explain normalization using functional dependencies, multi valued dependencies.	CLO-5	L2	7M

Unit-IV

8	a)	Discuss in detail about transaction processing mechanism	CLO-2	L3	7M
	b)	What are the different types of database recovery techniques with example.	CLO-3	L3	7M

(OR)

9	a)	Draw a state diagram and discuss the typical states that a transaction goes through during execution.	CLO-5	L2	7M
	b)	How to test the serializability of a schedule? Discuss with examples	CLO-5	L3	7M

a) List out the applications of DBMS.

- Railway And Airline Reservation System.
- Library Management System.
- Banking.
- Education Sector (Schools and Colleges)
- Social Media Sites (Instagram, Facebook, etc.)
- Online Shopping (E-commerce Platforms Like Amazon)
- Human Resource Management.
- Manufacturing.

b) Define single valued and multivalued attributes.

Single valued attributes consist of a single value for each entity instance and can't store more than one value. Multi-valued attributes can take up and store more than one value at a time for an entity instance from a set of possible values. They are represented by a co-centric elliptical shape.

c) What are the disadvantages of file processing system?

Disadvantages of File Processing System :

- Slow access time
- Presence of redundant data
- Inconsistent Data
- Data Integrity Problems
- Difficulty in recovery of corrupt data
- Lack of Atomicity
- Problem in Concurrent Access
- Unauthorized Access

d) Give the levels of data abstraction.

There are three levels of abstraction :

Physical Level: Defines how the data is actually stored employing various data structures.

Logical level: Describes the relationship which exists among the stored data.

View level: Provides a high-level view of a section of data.

e) Give the syntax and example for CREATE command in SQL.

The CREATE TABLE statement is used to create a new table in a database.

Syntax

```
CREATE TABLE table_name (
   column1 datatype,
   column2 datatype,
   column3 datatype,
  ....
);
```

f) Give an example for foreign key.

In the Student table, the field Stud_Id is a primary key because it is uniquely identifying all other fields of the Student table. On the other hand, Stud_Id is a foreign key attribute for the Department table because it is acting as a primary key attribute for the Student table.

g) What is Natural Join? Give an example.

A NATURAL JOIN is a JOIN operation that creates an implicit join clause for you based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables. A NATURAL JOIN can be an INNER join, a LEFT OUTER join, or a RIGHT OUTER join. The default is INNER join.

h) What is B+ tree?

A B+ tree is an advanced form of a self-balancing tree in which all the values are present in the leaf level.An important concept to be understood before learning B+ tree is multilevel indexing. In multilevel indexing, the index of indices is created as in figure below. It makes accessing the data easier and faster.

i) What are the properties of decomposition?

Whenever we decompose a relation, there are certain properties that must be satisfied to ensure no information is lost while decomposing the relations. These properties are: Lossless Join Decomposition. Dependency Preserving.

j) What is Multi value dependency? Give an example

Multivalued dependency (MVD) states that two independent data variable rows say 'B' and 'C' in a table can be dependent on the third variable row 'A' respectively, i.e., for same value of attribute 'A', we can have multiple values of component 'B' and 'C'.

k) What is the use of checkpoints in recoverability?

Checkpoints help us in recovering the transaction of the database in case of a random shutdown of the database. It enhancing the consistency of the database in case when multiple transactions are executing in the database simultaneously. It increasing the data recovery process

l) When the transaction will be rolled back?

A transaction cannot be rolled back after a COMMIT TRANSACTION statement is executed, except when the COMMIT TRANSACTION is associated with a nested transaction that is contained within the transaction being rolled back.

m) What is a Time stamp?

Timestamp is a unique identifier created by the DBMS to identify the relative starting time of a transaction. Typically, timestamp values are assigned in the order in which the transactions are submitted to the system. So, a timestamp can be thought of as the transaction start time.

n) Define strict schedule.

If a transaction is neither allowed to read nor write a data item in a schedule till the last transaction that has been written is committed or aborted, then such a schedule is called a Strict Schedule. In other words, a Strict Schedule allows only committed read and write operations

2 a **Describe Three schema architecture and Data independence in detail. 7M**

Description-3M

Diagram-4M

Many Web applications use an architecture called the **three-tier architecture**, which adds an intermediate layer between the client and the database server, as illustrated in Figure.

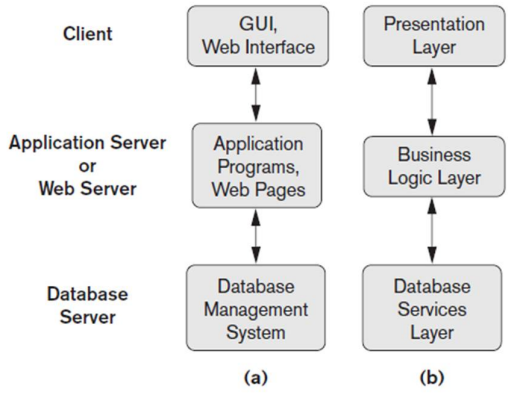


Fig: Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

This intermediate layer or **middle tier** is called the **application server** or the **Web server**, depending on the application. This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server. It can also improve database security by checking a client's credentials before forwarding a request to the database server. Clients contain user interfaces and Web browsers. The intermediate server accepts requests from the client, processes the request and sends database queries and commands to the database server, and then acts as a conduit for passing (partially) processed data from the database server to the clients, where it may be processed further and filtered to be presented to the users. Thus, the *user interface, application rules,* and *data access* act as the three tiers.

Data Independence The capacity to change the schema at one level of a database system without having to change the schema at the next higher level called **data independence.**

- We can define two types of data independence:
  1. **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs.
     - We may change the conceptual schema to **expand the database** (by adding a record type or data item), to change constraints, or to **reduce the database** (by removing a record type or data item).
  2. **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema.
     - Changes to the internal schema may be needed because some physical files were reorganized to improve the performance of retrieval or update.
     - If the same data as before remains in the database, we should not have to change the conceptual schema.

2b Define DBMS. List Database users and their roles 7M

- A database management system (DBMS) is a collection of programs that enables users to create and maintain a database.
- The DBMS is facilitates the processes of defining, constructing, manipulating, and sharing Databases among various users and applications
  - **Database Administrators**
  - **Database Designers**

1. **Database Administrators:**
   - The **database administrator (DBA)** is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.
   - The DBA is accountable for problems such as security breaches and poor system response time.

2. **Database Designers:**
   - **Database designers** are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data.
   - These tasks are undertaken before the database is actually implemented and populated with data.
   - It is the responsibility of database designers to communicate with all prospective database users in order to understand their requirements and to create a design that meets these requirements.

3. **End Users:**
   - **End users** are the people whose jobs require access to the database for querying, updating, and generating reports.
   - There are several categories of end users:
     - ✓ **Casual end users** occasionally access the database, but they may need different information each time.
     - ✓ **Naive** or **parametric end users** their main job function revolves around constantly querying and updating the database, using standard types of queries and updates(**canned transactions)** have been carefully programmed and tested.
     - ✓ **Sophisticated end users** include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.
     - ✓ **Standalone users** maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces.

4. **System Analysts and Application Programmers(Software Engineers):**
   - **System analysts** determine the requirements of end users, especially naive and parametric end users, and develop specifications for standard canned transactions that meet these requirements.
   - **Application programmers** implement these specifications as programs; then they test, debug, document, and maintain these canned transactions.
   - Such analysts and programmers commonly referred to as **software developers** or **software engineers**.

3 a Discuss in detail about the main steps in the database design and clearly focus in detail about the goal of each step? In which steps is the ER model mainly used? 7M
Description-4m
Explanation-3m

In general, the schema design process should be considered an iterative refinement process, where an initial design is created and then iteratively refined until the most suitable design is reached.

Some of the refinements that are often used include the following:

- A concept may be first modeled as an attribute and then refined into a relationship because it is determined that the attribute is a reference to another entity type. It is often the case that a pair of such attributes that are inverses of one another are refined into a binary relationship. It is important to note that in our notation, once an attribute is replaced by a relationship, the attribute itself should be removed from the entity type to

avoid duplication and redundancy.

- Similarly, an attribute that exists in several entity types may be elevated or promoted to an independent entity type. For example, suppose that several entity types in a UNIVERSITY database, such as STUDENT, INSTRUCTOR, and COURSE, each has an attribute Department in the initial design; the designer may then choose to create an entity type DEPARTMENT with a single attribute Dept_name and relate it to the three entity types (STUDENT, INSTRUCTOR, and COURSE) via appropriate relationships.
- An inverse refinement to the previous case may be applied—for example, if an entity type DEPARTMENT exists in the initial design with a single attribute Dept_name and is related to only one other entity type, STUDENT. In this case, DEPARTMENT may be reduced or demoted to an attribute of STUDENT.

3b) Design an ER Model for College Admission System. 7M

Description-4m
Relationships-3m

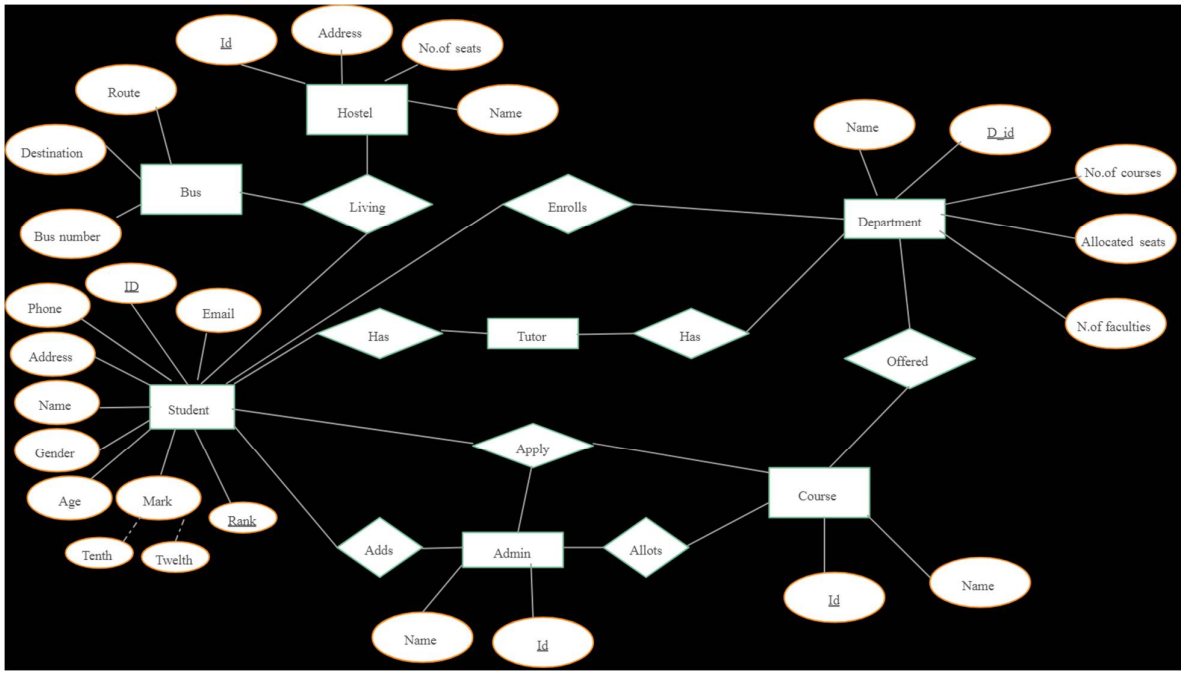
The admission system of a college can be represented using an Entity-Relationship (ER) diagram. The ER diagram is a graphical representation of the entities, relationships, and attributes involved in the system. In the above ER diagram, there are four main entities: Student, Course, Department, and Admission. These entities are represented by rectangles. **The relationships between these entities are represented by diamond-shaped symbols.**

1. **Student Entity**: The Student entity represents the students who apply for admission to the college. The entity has three attributes: StudentID, Name, and Address.
2. **Course Entity**: The Course entity represents the courses offered by the college. The entity has three attributes: CourseID, CourseName, and Credits.
3. **Department Entity**: The Department entity represents the different departments within the college. The entity has two attributes: DepartmentID and DepartmentName.
4. **Admission Entity**: The Admission entity represents the admission process for students. The entity has four attributes: AdmissionID, AdmissionDate, Status, and Fees.

The relationships between these entities are defined as follows:

1. One-to-Many Relationship between Department and Course: A department can offer multiple courses, but a course can only belong to one department. This relationship is represented by the arrow from Department to Course.
2. **Many-to-Many Relationship between Student and Course:** A student can take multiple courses, and a course can be taken by multiple students. This relationship is represented by the diamond-shaped symbol between Student and Course. This relationship is further expanded to include the Admission entity, which captures the details of a student's admission to a course.
3. **One-to-Many Relationship between Department and Admission:** A department can admit multiple students, but a student can only be admitted to one department. This relationship is represented by the arrow from Department to Admission.
4. **One-to-Many Relationship between Course and Admission:** A course can admit multiple students, but a student can only be admitted to one course. This relationship is represented by the arrow from Course to Admission.

Using the above ER diagram, it is possible to design a database schema for the admission system of a college. The schema can be further expanded to include additional entities and relationships as required by the specific needs of the college.



Unit-II

4a) Explain in detail about different types of attributes with diagrammatical representation in ER model. 7M

Description-4m

List of Attributes-3m

1.2.1 Entities and Attributes:

- An **entity** may be a **thing** or **object that can exist in the real world** (example, a person, a house, employee, a company, or a job).
- An **attribute** represents some **property that further describes an entity**, for example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job.
- A particular entity will have a value for each of its attributes.
- Figure 3.3 shows two entities and the values of their attributes.

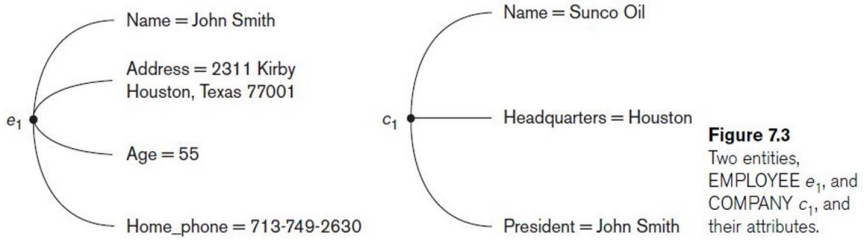


Figure 7.3
Two entities, EMPLOYEE e_1, and COMPANY c_1, and their attributes.

Figure 3.3 Two entities, EMPLOYEE e1, and COMPANY c1, and their attributes.

- Several types of attributes occur in the ER model:
  - **i)** ***simple versus composite***
  - **ii)** ***single valued versus multivalued***
  - **iii)** ***Stored versus derived.***
  - **iv)** ***NULL value***
  - **v)** ***Complex Attributes***

i) **Simple (Atomic) versus Composite Attributes:**

- An attribute **having only one part** is called simple (or atomic) attribute. For example, age of a person.
- An attribute can be **divided into smaller subparts** is called **Composite attributes**. For

subdivided into three simple component attributes: Number, Street, and Apartment_number, as shown in Figure 3.4.

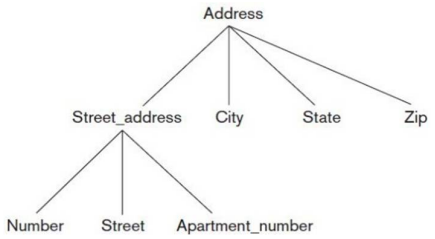


Figure 3.4 A hierarchy of composite attributes.

ii) **Single-Valued versus Multivalued Attributes:**

- An attributes have a single value for a particular entity is are called **single-valued**. For example, gender of employee.
- An attribute can have a set of values for the same entity called **multivalued.** For example a Colors of a car, or a College_degrees for a person.
- A multivalued attribute may **have lower and upper bounds** to constrain the ***number of values*** allowed for each individual entity.

iii) **Stored versus Derived Attributes:**

- In some cases, two (or more) attribute values are related. For example, the Age and Birth_date attributes of a person.
- For a particular person entity, the value of Age can be determined from the current (today's) date and the value of that person's Birth_date.
- The Age attribute is hence called a **derived attribute** and is said to be **derivable from** the Birth_date attribute, which is called a **stored attribute**.

iv) **NULL Values:**

- In some cases, a particular entity may not have an applicable value for an attribute that value is called **NULL**. For example middle name of a person.
- NULL can also be used if we **do not know**(or **unknow)** the value of an attribute for a particular entity
- The ***unknown*** category of NULL can be further classified into two cases.
  - ✓ The first case arises ***when it is known*** that the attribute value exists but is ***missing*** - for example, if the Height attribute of a person is listed as NULL.
  - ✓ The second case arises ***when it is not known*** whether the attribute value exists - for example, Minit of a person is NULL.

v) **Complex Attributes:**

- **Composite** and **multivalued** attributes can be nested arbitrarily.
- We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multivalued attributes between braces { }. Such attributes are called **complex attributes**.

```
{ Address_phone ( { Phone(Area_code, Phone_number) },
                  Address( Street_address ( Number, Street, Apartment_number, City,State,Zip) ) }
```

4b) What is aggregate function? Explain all aggregate functions with the help of example. **7M** L2

Description-4m
Syntax-3m

An aggregate function in SQL returns one value after calculating multiple values of a column. We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

There are 5 types of SQL aggregate functions:

Count()
Sum()
Avg()
Min() Max() COUNT() Function

The COUNT() function returns the number of rows in a database table.

Syntax:

COUNT(*)
or
COUNT([ALL|DISTINCT] expression)

SUM() Function

The SUM() function returns the total sum of a numeric column.

Syntax:

SUM()
or
SUM([ALL|DISTINCT] expression)

AVG() Function

The AVG() function calculates the average of a set of values.

Syntax:

AVG()
or
AVG([ALL|DISTINCT] expression)

MIN() Function

The MIN() aggregate function returns the lowest value (minimum) in a set of non-NULL values.

Syntax:

MIN()
or
MIN([ALL|DISTINCT] expression)

MAX() Function

The MAX() aggregate function returns the highest value (maximum) in a set of non-NULL values.

Syntax:

AVG()
or
AVG([ALL|DISTINCT] expression) MAX() Function

The MAX() aggregate function returns the highest value (maximum) in a set of non-NULL values.

Syntax:

AVG()
or
AVG([ALL|DISTINCT] expression)

MAX() Function

The MAX() aggregate function returns the highest value (maximum) in a set of non-NULL values.

Syntax:

AVG()
or
AVG([ALL|DISTINCT] expression)

MAX() Function

The MAX() aggregate function returns the highest value (maximum) in a set of non-NULL values.

Syntax:

AVG()
or
AVG([ALL|DISTINCT] expression)

(OR)

5a) State various fundamental operations in the relational algebra and explain them with suitable examples. 7M

Description-4m

Example-3m

- The basic set of operations for the relational model is the relational algebra.
- A sequence of relational algebra operations forms a relational algebra expression, whose result will also be a relation that represents the result of a database query (or retrieval request).
- The relational algebra is very important for several reasons.
    1. It provides a formal foundation for relational model operations.
    2. It is used as a basis for implementing and optimizing queries in the query processing and optimization modules that are integral parts of relational database management systems (RDBMSs).
    3. Third, some of its concepts are incorporated into the SQL standard query language for RDBMSs.
- The relational algebra operations can be divided into two groups.
    - One group includes set operations from mathematical set theory. Set operations include UNION, INTERSECTION, SET DIFFERENCE, and CARTESIAN PRODUCT (also known as CROSS PRODUCT).
    - The other group consists of operations developed specifically for relational databases—these include SELECT, PROJECT, and JOIN, among others

Unary Relational Operations: SELECT and PROJECT L3

- The SELECT and PROJECT operator are **unary**; that is, it is applied to a **single relation (table)**.

1.1.1 SELECT Operation:

- The SELECT operation is used to **retrieve a subset of the tuples from a relation that satisfies a selection condition**.
- **Syntax:**

$$\sigma_{<\text{selection condition}>}(R)$$

- Where the symbol **σ (sigma)** is used to denote the SELECT operator,
  selection condition is a Boolean expression (condition) specified on the attributes of relation R.
- For example, to select the EMPLOYEE tuples whose department is 4, SELECT operation as follows:

$$\sigma_{Dno=4}(EMPLOYEE)$$

PROJECT Operation:

- The PROJECT operation is used to **retrieve only certain attributes (columns) in a relation (table).**
- The PROJECT operation is display the attributes **in the same order as they appear in the list.**
- **Syntax:**

$$\pi_{<\text{attribute list}>}(R)$$

- Where **π** (**pi**) is the symbol used to represent the PROJECT operation,
  <attribute list> is the desired sub-list of attributes from the relation R.
- For example, to list each employee's first and last name and salary, we can use the PROJECT

operation as follows:

$\pi_{Lname, Fname, Salary}$(EMPLOYEE)

RENAME Operation

- To rename the attributes in a relation, we simply list the new attribute names in parentheses.
- Rename operation is denoted with the symbol ρ (rho)
- Example:

TEMP ← $\sigma_{Dno=5}$ (EMPLOYEE)

R (First_name, Last_name, Salary) ← $\pi_{Fname, Lname, Salary}$(TEMP)

The UNION, INTERSECTION, and MINUS Operations

- Set theoretic operations are used to merge the elements of two sets in various ways, including UNION, INTERSECTION, and SET DIFFERENCE (also called MINUS or EXCEPT).
- These are binary operations; that is, each is applied to two relations.
- The two relations on which any of these three operations are applied must follow two rules.
  - ✓ Two relations must have **the same number of attributes**.
  - ✓ Two relations must have **union compatibility** or **type compatibility (**i.e., same type of tuples**)**.
- Two relations R(A1 , A2 , ..., An) and S(B1 , B2 , ..., Bn) are said to be union compatible (or type compatible) if they have the same degree n and if dom(Ai) = dom(Bi).
- We can define the three operations **UNION**, **INTERSECTION**, and **SET DIFFERENCE** on two union-compatible relations R and S as follows:

  - ❖ **UNION:** The result of this operation, **denoted by R ∪ S**, is a relation that includes **all tuples that are either in R or in S or in both R and S**. **Duplicate tuples are eliminated**.

  - ❖ **INTERSECTION:** The result of this operation, **denoted by R ∩ S**, is a relation that includes **all tuples that are in both R and S**.

  - ❖ **SET DIFFERENCE (or MINUS):** The result of this operation, **denoted by R – S**, is a relation that includes **all tuples that are in R but not in S**.

  Figure: The set operations UNION, INTERSECTION, and MINUS.
  (a) Two union-compatible relations. (b) STUDENT ∪ INSTRUCTOR.
  (c) STUDENT ∩ INSTRUCTOR. (d) STUDENT − INSTRUCTOR.
  (e) INSTRUCTOR − STUDENT

5b) Explain in detail about different types of joint operations with example. 7M

Description-4m L2
Syntax-3m

The JOIN Operation

- The JOIN operation, denoted by ⋈, is used to combine related tuples from two relations into single "longer" tuples.
- There are several join operations: **THETA JOIN, EQUIJOIN, NATURAL JOIN**

THETA JOIN:

- A JOIN operation with general join condition is called a **THETA JOIN**.
- Syntax:

- where each <condition> is of the form A_i θ B_j, Ai is an attribute of R, B_j is an attribute of S, A_i and B_j have the same domain, and θ (theta) is one of the comparison operators {=, <, ≤, >, ≥, ≠}.
- Example: to retrieve the employee names and their grades

  $\pi_{Ename, Grade}(Emp \bowtie_{emp.sal>=salgrade.losal\ AND\ emp.sal>=salgrade.hisal\ Salgrade} Salgrade)$
- Tuples whose join attributes are NULL or for which the join condition is FALSE do not appear in the result.

EQUIJOIN:

- A JOIN, where the only comparison operator used is =, is called an EQUIJOIN.
- EQUIJOIN we **always have one or more pairs of attributes that have identical values in every tuple**.
- Example: $\pi_{Ename, Deptno, Dname}(Emp \bowtie_{emp.deptno=dept.deptno} Dept)$

NATURAL JOIN:

- NATURAL JOIN denoted by *****.
- A NATURAL JOIN requires that the two join **attributes** (or **each pair of join attributes**) **have the same name** in both relations.
- NATURAL JOIN was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition.
- Example: $\pi_{Ename, Deptno, Dname}(Emp * Dept)$
- If this is not the case, a renaming operation is applied first.

  Example: e1←EMP e2←EMP mgr←e2.Empno

  $\pi_{e1.Ename, e2.Ename}(e1 * e2)$
- **Join attribute:** an **attribute is used to join the both relations** is called join attribute.
- In JOIN, only combinations of tuples satisfying the join condition appear in the result, whereas in the CARTESIAN PRODUCT all combinations of tuples are included in the result.

The expected size of the join result divided by the maximum size $n_R * n_S$ leads to a ratio called **join selectivity**, which is a property of each join condition.

Unit-III

6a) List and explain various functional dependency mechanism.
List-3m
Description-4m

1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Multivalued functional dependency
4. Transitive functional dependency

1. Trivial Functional Dependency

In **Trivial Functional Dependency**, a dependent is always a subset of the determinant.i.e. If **X → Y** and **Y is the subset of X**, then it is called trivial functional dependency

roll_no	name	age
43	pqr	18
44	xyz	18

Here, **{roll_no, name} → name** is a trivial functional dependency, since the dependent **name** is a subset of determinant set **{roll_no, name}.** Similarly, **roll_no → roll_no** is also an example of trivial functional dependency.

2. Non-trivial Functional Dependency

In **Non-trivial functional dependency**, the dependent is strictly not a subset of the determinant. i.e. If

roll_no. Similarly, **{roll_no, name} → age** is also a non-trivial functional dependency, since **age** is **not a subset of {roll_no,name}**

3. Multivalued Functional Dependency

In **Multivalued functional dependency**, entities of the dependent set are **not dependent on each other.** i.e. If **a → {b, c}** and there exists **no functional dependency** between **b and c**, then it is called a **multivalued functional dependency.** Here, **roll_no → {name, age}** is a multivalued functional dependency, since the dependents **name** & **age** are **not dependent** on each other(i.e. **name → age** or **age → name doesn't exist !**)

4. Transitive Functional Dependency

In transitive functional dependency, dependent is indirectly dependent on determinant. i.e. If **a → b** & **b → c**, then according to axiom of transitivity, **a → c**. This is a **transitive functional dependency.**

Here, **enrol_no → dept** and **dept → building_no.** Hence, according to the axiom of transitivity, **enrol_no → building_no** is a valid functional dependency. This is an indirect functional dependency, hence called Transitive functional dependency.

5. Fully Functional Dependency

In full functional dependency an attribute or a set of attributes uniquely determines another attribute or set of attributes. If a relation R has attributes X, Y, Z with the dependencies X->Y and X->Z which states that those dependencies are fully functional..

6. Partial Functional Dependency

In partial functional dependency a non key attribute depends on a part of the composite key, rather than the whole key. If a relation R has attributes X, Y, Z where X and Y are the composite key and Z is non key attribute. Then X->Z is a partialfunctional dependency in RBDMS.

6b) Given a relation with attributes R= (A,B,C,D,E,F,G,H) and following functional dependencies. 7M L5

CH→G
A→BC
B→CFG
E→A
F→EG

State the strongest normal form that the relation R is in.
If R is not in BCNF, decompose it into a collection of BCNF relations.

Description-4m

Explanation-3m

Given a relation with attributes R= (A,B,C,D,E,F,G,H) and following functional dependencies.
CH→G
A→BC
B→CFG
E→A
F→EG

Since D is not part of any functional dependency so it can be a candidate key or maybe part of a candidate key.

To find more candidate keys add A, B, C, D, E, G, and H to D & find its **closure**:

$(AD)^+ = \{ABCDEFGH\}$

$(BD)^+ = \{ABCDEFGH\}$

$(CD)^+ = \{CD\}$

$(ED)^+ = \{ABCDEFGH\}$

$(FD)^+ = \{ABCDEFGH\}$

$(GD)^+ = \{GD\}$

$(HD)^+ = \{HD\}$

Since **AD, BD, ED & FD** gives all attributes, so they are candidate keys.

But the dependencies,

A → BC, B → CFH, and F → EG are partial dependencies.

{Here **C, G, H** are non-key attributes.}

Hence, the given relation is in 1NF but not in 2NF.

(OR)

7 a **Construct B tree to insert the following key values (order of the tree is 4)** 7M
10,50,5,7,2,345,55,22,44,56,58,70,17,6,21
Description-4m
Example-3m

To construct a B-tree with an order of 4 and insert the given key values, follow these steps:

Start with an empty B-tree.
Insert each key value one by one, maintaining the B-tree properties at each step.
B-tree properties for order m:
a. Every node has at most m children.
b. Every node (except the root) has at least m/2 children.
c. The root has at least 2 children if it's not a leaf.
d. All leaves are at the same level.

Let's go through the insertion process:

Insert 10:

```
10
```

Insert 50:

```
10, 50
```

Insert 5:

```
  10, 50
  /   \
 5
```

Insert 7 (Causes a split):

```
  10, 50
  /   \
 5    7
```

Insert 2 (Causes a split):

```
   7
  / \
 2   5

10, 50
```

Insert 345 (Causes a split):

```
   7
  / \
 2   5

10, 50, 345
```

Insert 55:

```
   7
  / \
 2   5

10, 50, 55, 345
```

Insert 22:

```
      7, 22
     / | \
    2  5  10, 50, 55, 345
```

Insert 44 (Causes a split):

```
      7, 22
     / | \
    2  5  10, 50, 55, 345, 44
```

Insert 56:

```
      7, 22
     / | \ \
    2  5  10, 50, 55, 345, 44, 56
```

Insert 58:

```
      7, 22, 56
     / | \ \ \
    2  5  10, 50, 55, 58, 345, 44
```

Insert 70 (Causes a split):

```
      7, 22, 56
     / | \ \ \
    2  5  10, 50, 55, 58, 70, 345, 44
```

Insert 17:

```
      7, 17, 22, 56
     / | \ \ \
    2  5  10, 50, 55, 58, 70, 345, 44
```

Insert 6:

```
      7, 17, 22, 56
     /  |  \  \
    2  5, 6 10, 50, 55, 58, 70, 345, 44
```

Insert 21:

```
      7, 17, 22, 56
     /  |  \  \
    2  5, 6 10, 21, 50, 55, 58, 70, 345, 44
```

The final B-tree with order 4 and the given key values is:

```
       7, 17, 22, 56
      /  |  \  \
     2  5, 6 10, 21, 50, 55, 58, 70, 345, 44
```

7b **What is meant by normalization? Is it mandatory in a relational database design? Briefly explain normalization using functional dependencies, multi valued dependencies.** 7M

Description-4m

Example-3m

Normalization is the process of organizing a database to reduce redundancy and dependency. It is important because it helps to eliminate data inconsistencies and ensures that the data is stored in a logical and organized way.

During the normalization process of database design, make sure that proposed entities meet required normal form before table structures are created. Many real-world databases have been improperly designed or

burdened with anomalies if improperly modified during the course of time. You may be asked to redesign and modify existing databases. This can be a large undertaking if the tables are not properly normalized.

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

Example: Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF_YEAR are dependent on BIKE_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE_MODEL. The representation of these dependencies is shown below:

1. BIKE_MODEL → → MANUF_YEAR
2. BIKE_MODEL → → COLOR

This can be read as "BIKE_MODEL multidetermined MANUF_YEAR" and "BIKE_MODEL multidetermined COLOR".

8 a **Discuss in detail about transaction processing mechanism** **7M**
Description-4m
Example-3m

- The transaction is a set of logically related operation. It contains a group of tasks.
- A transaction is an action or series of actions. It is performed by a single user to perform operations for accessing the contents of the database.

Example: Suppose an employee of bank transfers Rs 800 from X's account to Y's account. This small transaction contains several low-level tasks:

X's Account

Open_Account(X)

Old_Balance = X.balance

New_Balance = Old_Balance - 800

X.balance = New_Balance

Close_Account(X)

Y's Account

Open_Account(Y)

Old_Balance = Y.balance

New_Balance = Old_Balance + 800

Y.balance = New_Balance

Close_Account(Y)

Operations of Transaction:

Following are the main operations of transaction:
Read(X): Read operation is used to read the value of X from the database and stores it in a buffer in main memory.
Write(X): Write operation is used to write the value back to the database from the buffer.
Let's take an example to debit transaction from an account which consists of following operations:

1. R(X);
2. X = X - 500;
3. W(X);

Let's assume the value of X before starting of the transaction is 4000.
The first operation reads X's value from database and stores it in a buffer.
The second operation will decrease the value of X by 500. So buffer will contain 3500.
The third operation will write the buffer's value to the database. So X's final value will be 3500.
But it may be possible that because of the failure of hardware, software or power, etc. that transaction may fail before finished all the operations in the set.

For example: If in the above transaction, the debit transaction fails after executing operation 2 then X's value will remain 4000 in the database which is not acceptable by the bank.
To solve this problem, we have two important operations:
Commit: It is used to save the work done permanently.

8b) What are the different types of database recovery techniques with example? 7M
Description-4m
Example-3m

Database recovery techniques are used in database management systems (DBMS) to restore a database to a consistent state after a failure or error has occurred. The main goal of recovery techniques is to ensure data integrity and consistency and prevent data loss. There are mainly two types of recovery techniques used in DBMS:

Rollback/Undo Recovery Technique: The rollback/undo recovery technique is based on the principle of backing out or undoing the effects of a transaction that has not completed successfully due to a system failure or error. This technique is accomplished by undoing the changes made by the transaction using the log records stored in the transaction log. The transaction log contains a record of all the transactions that have been performed on the database. The system uses the log records to undo the changes made by the failed transaction and restore the database to its previous state.

Commit/Redo Recovery Technique: The commit/redo recovery technique is based on the principle of reapplying the changes made by a transaction that has been completed successfully to the database. This technique is accomplished by using the log records stored in the transaction log to redo the changes made by the transaction that was in progress at the time of the failure or error. The system uses the log records to reapply the changes made by the transaction and restore the database to its most recent consistent state.

In addition to these two techniques, there is also a third technique called checkpoint recovery. Checkpoint recovery is a technique used to reduce the recovery time by periodically saving the state of the database in a checkpoint file. In the event of a failure, the system can use the checkpoint file to restore the database to the most recent consistent state before the failure occurred, rather than going through the entire log to recover the database.

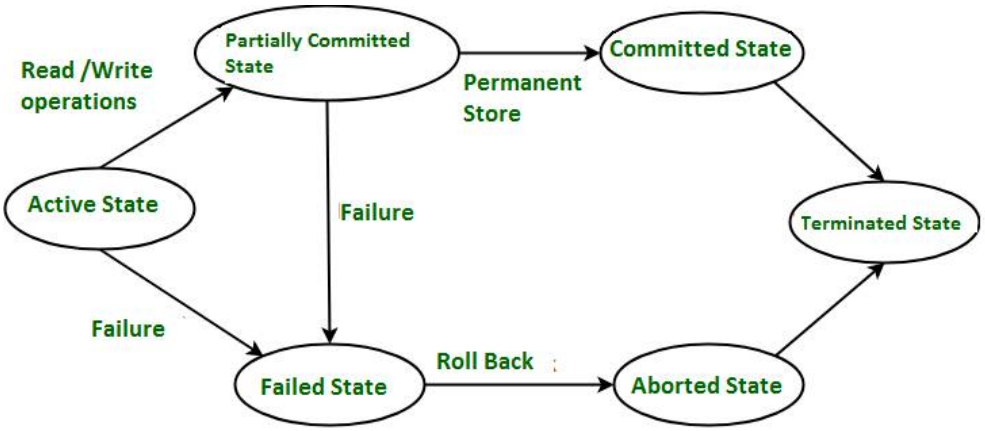
9 a **Draw a state diagram and discuss the typical states that a transaction goes through during execution.** **7M**

Description-4
Explanation-3m

States through which a transaction goes during its lifetime. These are the states which tell about the current state of the Transaction and also tell how we will further do the processing in the transactions. These states govern the rules which decide the fate of the transaction whether it will commit or abort.

They also use Transaction log. Transaction log is a file maintain by recovery management component to record all the activities of the transaction. After commit is done transaction log file is removed.

CLO-5 L2



Transaction States in DBMS

These are different types of Transaction States :

Active State –
When the instructions of the transaction are running then the transaction is in active state. If all the ‘read and write’ operations are performed without any error then it goes to the “partially committed state”; if any instruction fails, it goes to the “failed state”.

Partially Committed –
After completion of all the read and write operation the changes are made in main memory or local buffer. If the changes are made permanent on the DataBase then the state will change to “committed state” and in case of failure it will go to the “failed state”.

Failed State –
When any instruction of the transaction fails, it goes to the “failed state” or if failure occurs in making a permanent change of data on Data Base.

Aborted State –

After having any type of failure the transaction goes from “failed state” to “aborted state” and since in previous states, the changes are only made to local buffer or main memory and hence these changes are deleted or rolled-back.

Committed State –
It is the state when the changes are made permanent on the Data Base and the transaction is complete and therefore terminated in the “terminated state”.

Terminated State –
If there isn’t any roll-back or the transaction comes from the “committed state”, then the system is consistent and ready for new transaction and the old transaction is terminated.

9b) How to test the serializability of a schedule? Discuss with examples 7M
Description-4m
Example-3m

Serializability in DBMS guarantees that the execution of multiple transactions in parallel does not produce any unexpected or incorrect results. This is accomplished by enforcing a set of rules that ensure that each transaction is executed as if it were the only transaction running in the system.

A schedule can be represented as a set of transaction sequences, where each sequence corresponds to the operations performed by a single transaction. There are mainly two types of schedules.

Serial Schedule

Non-Serial Schedule

Testing of Serializability in DBMS

Testing for serializability in a database management system (DBMS) is an important step in ensuring that concurrent transactions executing in the database do not produce inconsistent or incorrect results. Serializability testing involves verifying that a given schedule of transactions is serializable, meaning that the effects of running the transactions concurrently are equivalent to running them serially, one after the other.

We can use below two techniques to test serializability in DBMS:

Serialization Graph

Precedence Graph

It can be described as a Graph G(V, E) with vertices V = "V1, V2, V3,…, Vn" and directed edges E = "E1, E2, E3,…, En". One of the two operations—READ or WRITE—performed by a certain transaction is contained in the collection of edges.

The meaning of the above graph is that, before Transaction y , Transaction x is performing a read or write operation.

Signature of the internal Examiner (B. Krishnaiah)

Name of the external Examiners	Name of the college	Dept.	Signature