CBCS SCHEME

USN | | | | | | | | | |

BCS755A

Seventh Semester B.E./B.Tech. Degree Examination, Dec.2025/Jan.2026
Introduction to DBMS

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
Q.1	a.	Define Database. List and briefly explain the advantages of using DBMS approach.	10	L2	CO1
	b.	Define DBMS. Discuss the characteristics of the database approach.	5	L2	CO1
	c.	With neat diagram, explain the three schema architecture.	5	L2	CO1
		OR			
Q.2	a.	Explain the component module of DBMS and their interactions with the help of neat diagram.	10	L2	CO1
	b.	Define the following terms: i) Weak entity ii) DBMS catalog iii) Value sets iv) Cardinality ratio v) Degree of a relationship	10	L2	CO1
		Module – 2			
Q.3	a.	Build an ER diagram of company system taking into account at least four entities. Indicate all keys, constraints and assumptions that are made.	10	L3	CO2
	b.	With an example, explain the steps of ER to relational mapping algorithm.	10	L2	CO2
		OR			
Q.4	a.	Build an ER diagram for university database by considering atleast 5 entities.	10	L3	CO2
	b.	Illustrate specialization and generalization with example.	10	L2	CO2

		Module – 3			
Q.5	**a.**	Explain about relational model constraints.	**10**	**L2**	**CO3**
	b.	By refereeing the following database schema: EMPLOYEE (Fname, Minit, Lname, SSN, Bdate, Address, Sex, Salary, Sup-SSN, Dno) Department (Dname, Dnumber, Mgr-SSN, Mgr-start-date) Dept-Locations (Dnumber, Dlocation) Project (Pname, Pnumber, Plocation, Dnum) Works – on (Essn, Pno, Hours) Dependent (Essn, Dependent-name, sex, Bdate, Relationship) Show the relational algebra expressions for the following queries. i) Retrieve the name and address of all employees who work for the 'Research' department. ii) Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' either as a worker or as a manager of the department that controls the project. iii) List the names of managers who have at least one dependent. iv) Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary. v) For each project, retrieve the project number, the project name, and the number of employees who work on that project.	**10**	**L3**	**CO3**
		OR			
Q.6	**a.**	Explain how the basic operations deal with constraints violations with example.	**10**	**L2**	**CO3**
	b.	Explain Division operation with example.	**10**	**L2**	**CO3**
		Module – 4			
Q.7	**a.**	Explain INSERT, DELETE, UPDATE statements in SQL taking suitable examples.	**10**	**L2**	**CO3**
	b.	What is Normalization? Explain 1NF, 2NF and 3NF with examples.	**10**	**L2**	**CO4**

		OR			
Q.8	**a.**	List and explain informal design guidelines for relation schemas.	**10**	**L2**	**CO4**
	b.	By refereeing the following database schema: Employee (Fname, Minit, Lname, SSN, Bdate, Address, Sex, Salary, Sup-SSN, Dno) Department (Dname, Dnumber, Mgr-SSN, Mgr-start-date) Dept-Locations (Dnumber, Dlocations) Project (Pname, Pnumber, Plocation, Dnum) Works-on (Essn, Pno, Hours) Dependent (Essn, Dependent-Name, Sex, Bdate, Relationship) Write queries in SQL i) Retrieve all the employee names who are working for department number 5. ii) Retrieve all the projects which are controlled by department number 4 iii) Retrieve the names of employees who have no dependents iv) Retrieve the employee name who is working on all the projects in which 'John Smith' works on. v) Retrieve all the project numbers along with number of employee working on each project.	**10**	**L3**	**CO3**
		Module – 5			
Q.9	**a.**	How are triggers and assertions defined in SQL? Explain.	**10**	**L2**	**CO3**
	b.	Discuss the Two-Phase locking techniques for concurrency control.	**10**	**L2**	**CO5**
		OR			
Q.10	**a.**	Explain views in SQL with example.	**10**	**L2**	**CO3**
	b.	Explain validation concurrency control techniques in databases.	**10**	**L2**	**CO5**

* * * * *

INTRODUCTION TO DBMS BCS755A
SEVENTH SEMESTER OPEN ELECTIVE-ECE DEC2025-JAN2026

1. a. Define Database. List and briefly explain the advantages of using DBMS approach. (10 Marks)

A **database management system** (DBMS) is a collection of programs enabling users to create and maintain a database.More specifically, a DBMS is a *general purpose* software system facilitating each of the following (with respect to a database):

- **definition:** specifying data types (and other constraints to which the data must conform) and data organization
- **construction:** the process of storing the data on some medium (e.g., magnetic disk) that is controlled by the DBMS
- **manipulation:** querying, updating, report generation
- **sharing:** allowing multiple users and programs to access the database "simultaneously"
- **system protection:** preventing database from becoming corrupted when hardware or software failures occur
- **security protection:** preventing unauthorized or malicious access to database.

ADVANTAGES OF USING DBMS:

i. Controlling Redundancy • redundancy in storing the same data multiple times leads to several problems o there is the need to perform a single logical update o storage space is wasted when the same data is stored repeatedly o Files that represent the same data may become inconsistent

ii. Restricting Unauthorized Access • A DBMS should provide a security and authorization subsystem • most users will not be authorized to access all information in the database

iii. Providing Persistent Storage for Program Objects • Databases can be used to provide persistent storage for program objects and data structures

iv. Providing Storage Structures and Search Techniques for Efficient Query Processing • the DBMS must provide specialized data structures and search techniques to speed up disk search for the desired records – index • The DBMS often has a buffering or caching module that maintains parts of the database in main memory buffers • The query processing and optimization module of the DBMS is responsible for choosing an efficient query execution plan for each query based on the existing storage structures.

v. Providing Backup and Recovery

responsible for recovery • the database is restored to the state it was in before the transaction started executing

vi. Providing Multiple User Interfaces • a DBMS should provide a variety of user interfaces o query languages for casual users, o programming language interfaces for application programmers, o forms and command codes for parametric users, and o menu-driven interfaces and natural language interfaces for standalone users

vii. Representing Complex Relationships among Data • A DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficientl

viii. Enforcing Integrity Constraints • The simplest type of integrity constraint involves specifying a data type for each data item • referential integrity constraint: specifying that a record in one file must be related to records in other files • Primary key constraint: uniqueness on data item values.

ix. Permitting inferencing and Actions Using Rules: A trigger is a form of a rule activated by updates to the table, which results in performing some additional operations to some other tables, sending messages, and so on.

x. Additional Implications of Using the Database Approach

Potential for Enforcing Standards: The DBA can enforce standards in a centralized database environment o Reduced Application Development Time: Development time using a DBMS is estimated to be one-sixth to one-fourth of that for a traditional file system o Flexibility: Modern DBMSs allow certain types of evolutionary changes to the structure of the database without affecting the stored data and the existing application programs o Availability of Up-to-Date Information: As soon as one user's update is applied to the database, all other users can immediately see this update o Economies of Scale: reduce the wasteful overlap activities thereby reducing the overall costs of operation and management.

1. b. Define DBMS. Discuss the characteristics of the database approach. (5 Marks)

A database management system (DBMS) is a collection of programs enabling users to create and maintain a database.

More specifically, a DBMS is a *general purpose* software system facilitating each of the following (with respect to a database):

- definition: specifying data types (and other constraints to which the data must conform) and data organization
- construction: the process of storing the data on some medium (e.g., magnetic disk) that is controlled by the DBMS
- manipulation: querying, updating, report generation
- sharing: allowing multiple users and programs to access the database "simultaneously"

- system protection: preventing database from becoming corrupted when hardware or software failures occur
- security protection: preventing unauthorized or malicious access to database.

Characteristics of DBMS:

Main Characteristics of database approach:

1. **Self-Description:** A database system includes —in addition to the data stored that is of relevance to the organization— a complete definition/description of the database's structure and constraints. This meta-data (i.e., data about data) is stored in the so-called system catalog, which contains a description of the structure of each file, the type and storage format of each field, and the various constraints on the data (i.e., conditions that the data must satisfy).

The system catalog is used not only by users (e.g., who need to know the names of tables and attributes, and sometimes data type information and other things), but also by the DBMS software, which certainly needs to "know" how the data is structured/organized in order to interpret it in a manner consistent with that structure. Recall that a DBMS is *general purpose*, as opposed to being a specific database application. Hence, the structure of the data cannot be "hard-coded" in its programs (such as is the case in typical *file processing* approaches), but rather must be treated as a "parameter" in some sense.

2. **Insulation between Programs and Data;** Data Abstraction:Program-Data Independence: In traditional file processing, the structure of the data files accessed by an application is "hard-coded" in its source code. (E.g., Consider a file descriptor in a COBOL program: it gives a detailed description of the layout of the records in a file by describing, for each field, how many bytes it occupies.)

If, for some reason, we decide to change the structure of the data (e.g., by adding the first two digits to the YEAR field, in order to make the program Y2K compliant!), every application in which a description of that file's structure is hard-coded must be changed!

In contrast, DBMS access programs, in most cases, do not require such changes, because the structure of the data is described (in the system catalog) separately from the programs that access it and those programs consult the catalog in order to ascertain the structure of the data (i.e., providing a means by which to determine boundaries between records and between fields within records) so that they interpret that data properly.

In other words, the DBMS provides a conceptual or logical view of the data to application programs, so that the underlying implementation may be changed without the programs being modified. (This is referred to as *program-data independence*.)

Also, which access paths (e.g., indexes) exist are listed in the catalog, helping the DBMS to determine the most efficient way to search for items in response to a query.

Note: In fairness to COBOL, it should be pointed out that it has a COPY feature that allows different application programs to make use of the same file descriptor stored in a "library". This provides some degree of program-data independence, but not nearly as much as a good DBMS does. *End of note.*

3. **Multiple Views of Data**: Different users (e.g., in different departments of an organization) have different "views" or perspectives on the database. For example, from the point of view of a Bursar's Office employee, student data does not include anything about which courses were taken or which grades were earned. (This is an example of a subset view.)

As another example, a Registrar's Office employee might think that GPA is a field of data in each student's record. In reality, the underlying database might calculate that value each time it is needed. This is called virtual (or derived) data.

A view designed for an academic advisor might give the appearance that the data is structured to point out the prerequisites of each course.

4. **Data Sharing and Multi-user Transaction Processing:** As you learned about (or will) in the OS course, the simultaneous access of computer resources by multiple users/processes is a major source of complexity. The same is true for multi-user DBMS's.Arising from this is the need for concurrency control, which is supposed to ensure that several users trying to update the same data do so in a "controlled" manner so that the results of the updates are as though they were done in some sequential order (rather than interleaved, which could result in data being incorrect).

This gives rise to the concept of a transaction, which is a process that makes one or more accesses to a database and which must have the appearance of executing in *isolation* from all other transactions (even ones that access the same data at the "same time") and of being *atomic* (in the sense that, if the system crashes in the middle of its execution, the database contents must be as though it did not execute at all).

Applications such as airline reservation systems are known as online transaction processing applications.

1. c. With a diagram, explain the three schema architectures. (5 Marks)

In 3 schema architecture, schemas can be defined at the following three levels:

1. The internal level has an internal schema, -describes the physical storage structure of the database. -uses a physical data model and describes the complete details of data storage and access paths for the database.

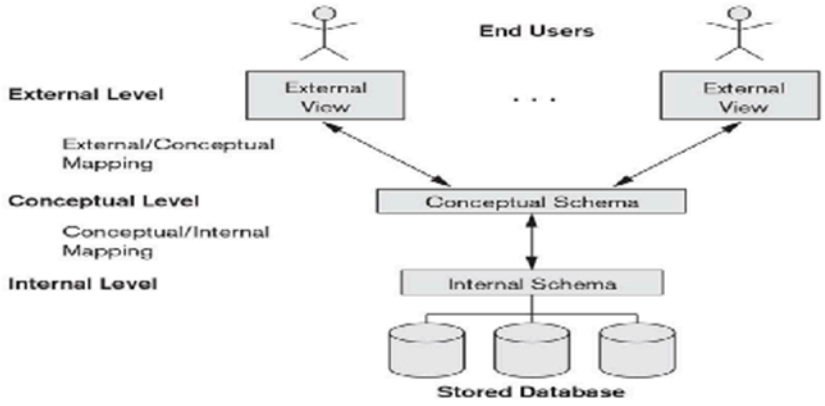
2. The conceptual level has a conceptual schema, -describes the structure of the whole database for a community of users -hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints -representational data model is used to describe the conceptual schema when a database system is implemented -This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

3. The external or view level includes a number of external schemas or user views, -describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. -As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

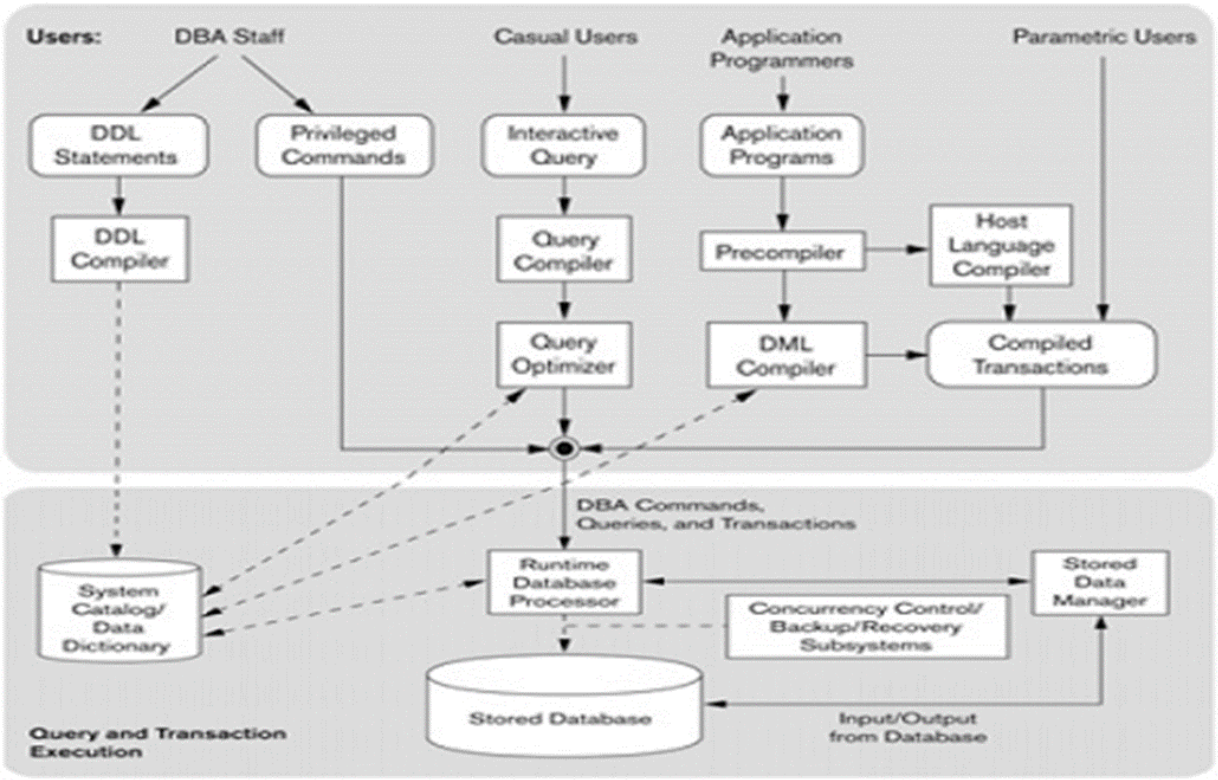
The DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are calledmappings. These mappings may be time-consuming, so some DBMSs—especially those that are meant to support small databases—do not support external views. Even in such systems, however, a certain amount of mapping is necessary to transform requests between the conceptual and internal levels.

The Three-Schema Architecture

The goal of the three-schema architecture is to separate the user applications from the physical database.



2. a. Explain the component module of DBMS and their interactions with the help of neat diagrams. (10 Marks)



A DBMS is a complex software system • discuss the types of software components that constitute a DBMS and the types of computer system software with which the DBMS interacts • two parts: o The top part: various users and their interfaces o The lower part: storage of data and processing of transactions • Access to the disk is controlled primarily by the operating system (OS) • buffer management module to schedule disk read/write, because this has a considerable effect on performance • A higher-level stored data manager module: controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog • DDL Statement: used by the DBA for defining the database and tuning it • DDL Compiler: processes schema definitions and stores descriptions of the schemas in the DBMS catalog.

Privileged commands: The commands which are exclusively used by the DBA • Interactive query: The interface by which the casual users and persons with occasional need for information from the database interact • Query compiler: Queries are parsed and validated for correctness of the query syntax • Query optimizer: Responsible for optimising the query and its execution.

Pre-compiler: Extracts DML commands from an application program written in a host programming language • DML compiler: Compilation of pre-compiled DMLC commands into object code for database access

Host language compiler: The rest of the program is sent to the host language compiler. • Compiled transactions: Canned transactions are executed repeatedly by parametric users, who simply supply the parameters to the transactions.

2. b. Define the following terms: Weak entity, DBMS catalog, Value sets, Cardinality ratio, Degree of a relationship. (10 Marks)

Weak Entity

A weak entity is an entity that does not have a primary key of its own and is identified using a partial key along with the primary key of a related strong entity.

DBMS Catalog

The DBMS catalog is a collection of metadata that stores information about the database structure, such as tables, attributes, data types, constraints, users, and access privileges. It is also called the data dictionary.**Value Sets**

A value set is the set of all possible values that an attribute can take. It defines the domain or permissible range of values for that attribute.

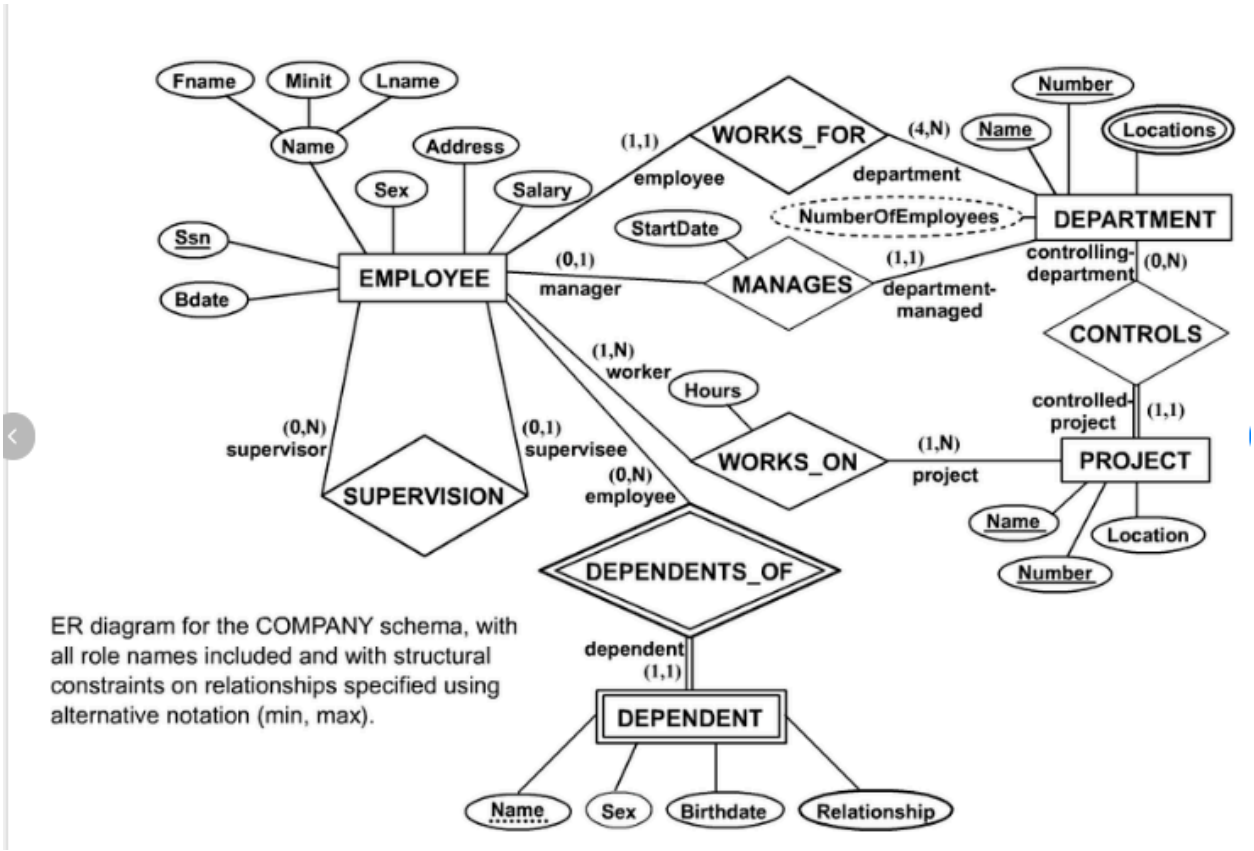
Cardinality Ratio

The cardinality ratio specifies the maximum number of entity instances that can participate in a relationship. Common cardinalities are one-to-one (1:1), one-to-many (1:N), and many-to-many (M:N).

Degree of a Relationship

The degree of a relationship indicates the number of entity types participating in a relationship. For example, unary (1), binary (2), and ternary (3) relationships.

3. a. Build an ER diagram of the company system taking into account at least four entities. Indicate all keys, constraints and assumptions that are made. (10 Marks)



ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).

3. b.With an example, explain the steps of ER to relational mapping algorithm. (10 Marks)

Map strong entity types to relations.
Map weak entity types to relations.
Map binary 1:1 relationships.
Map binary 1:N relationships.
Map binary M:N relationships.
Map multivalued attributes.
Map n-ary (ternary and higher) relationships.

Example ER Model

Consider a **University database** with:

- **Student** (StudentID, Name, Age)
- **Department** (DeptID, DeptName)

- **Course** (CourseID, CourseName)
- Relationships:
    - Student **belongs to** Department (1:N)
    - Student **enrolls in** Course (M:N)

Steps of ER to Relational Mapping

Step 1: Mapping Strong Entity Types

For each strong entity, create a relation with all simple attributes.
The primary key of the entity becomes the primary key of the relation.

Example:

- STUDENT (**StudentID**, Name, Age)
- DEPARTMENT (**DeptID**, DeptName)
- COURSE (**CourseID**, CourseName)

Step 2: Mapping Weak Entity Types

Create a relation for the weak entity including its attributes and the primary key of the owning strong entity.
The primary key is a combination of both.

Example:
If **DEPENDENT** is a weak entity of **EMPLOYEE**:

- DEPENDENT (**EmpID**, DependentName, Age)

Step 3: Mapping Binary 1:1 Relationships

Add the primary key of one entity as a foreign key in the other entity.
Choose the entity with total participation or fewer attributes.

Example:
If PERSON and PASSPORT have a 1:1 relationship:

- PASSPORT (**PassportNo**, IssueDate, PersonID)

Step 4: Mapping Binary 1:N Relationships

Add the primary key of the **1-side** entity as a foreign key in the **N-side** entity.

Example:
A Department has many Students:

- STUDENT (**StudentID**, Name, Age, DeptID)

(DeptID is a foreign key referencing DEPARTMENT)

Step 5: Mapping Binary M:N Relationships

Create a new relation whose primary key is a combination of the primary keys of the participating entities.
Include any relationship attributes.

Example:
Student enrolls in Course:

- ENROLLMENT (**StudentID**, **CourseID**, Semester, Grade)

Step 6: Mapping Multivalued Attributes

Create a separate relation for each multivalued attribute.
The primary key includes the entity key and the attribute.

Example:
Student has multiple phone numbers:

- STUDENT_PHONE (**StudentID**, PhoneNumber)

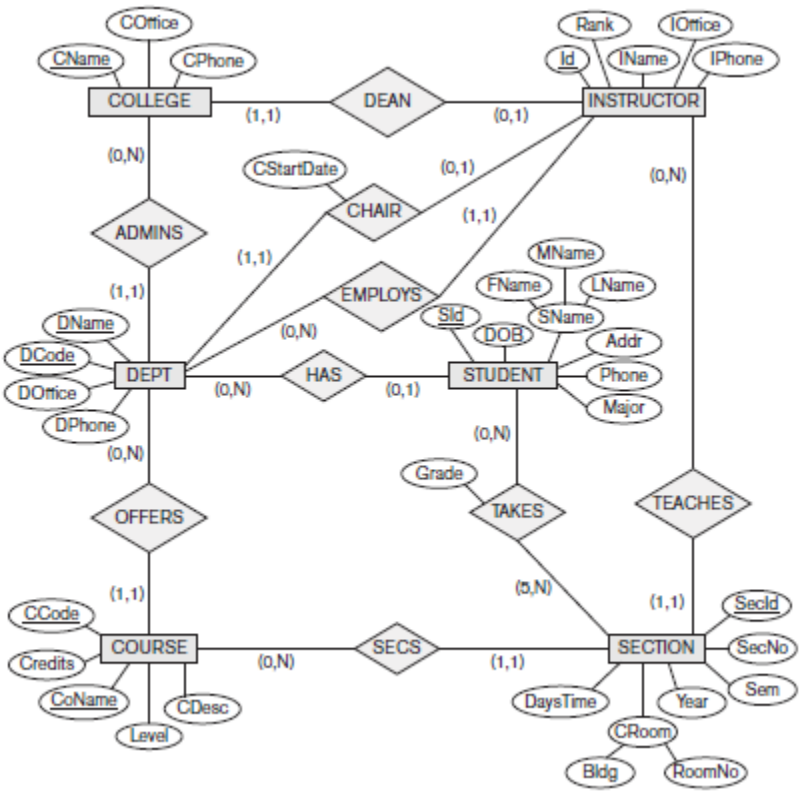
Step 7: Mapping N-ary Relationships

Create a new relation including the primary keys of all participating entities and any relationship attributes.

Example:
SUPPLY (Supplier, Part, Project):

- SUPPLY (**SupplierID**, **PartID**, **ProjectID**, Quantity)

4. a. Build an ER diagram for university database by considering at least 5 entities. (10 Marks)



4. b. Illustrate specialization and generalization with examples. (10 Marks)

Specialization is a top-down approach in which a higher-level entity set is divided into lower-level entity sets based on distinguishing characteristics.

Example

Consider an EMPLOYEE entity. Employees can be classified into TEACHING and NON-TEACHING staff.

- EMPLOYEE (EmpID, Name, Salary)
- TEACHING (Subject, Qualification)
- NON-TEACHING (Role, Experience)

Here, TEACHING and NON-TEACHING are specialized entities derived from EMPLOYEE.

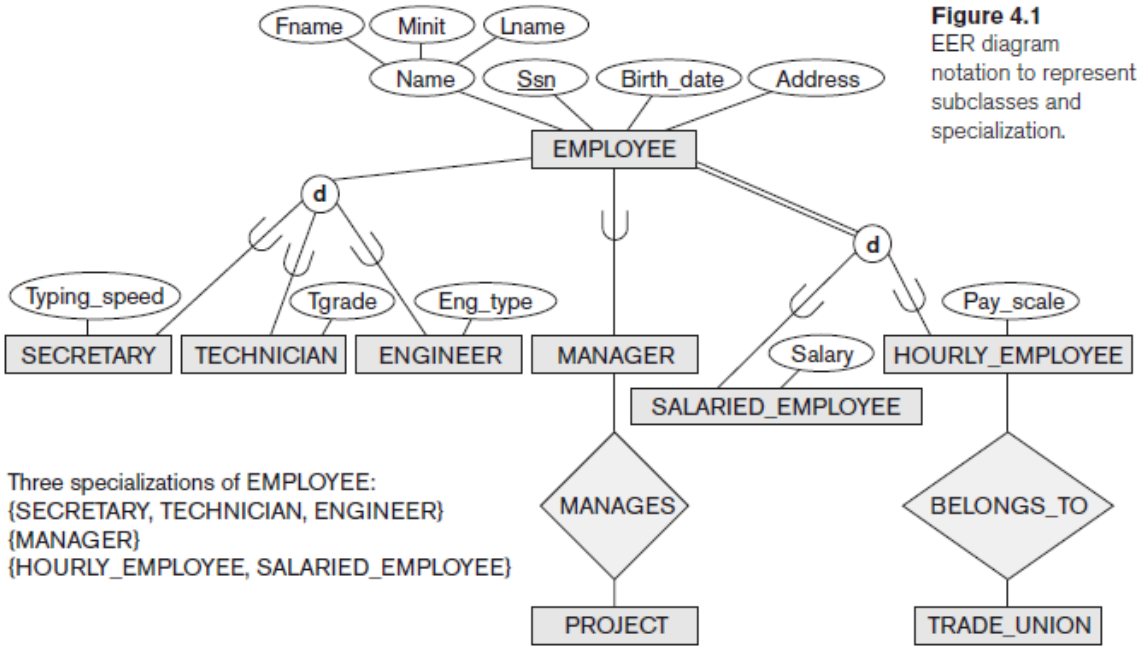


Figure 4.1
EER diagram notation to represent subclasses and specialization.

Generalization

Definition

Generalization is a **bottom-up approach** where multiple lower-level entities are combined into a **higher-level entity** by identifying common features.

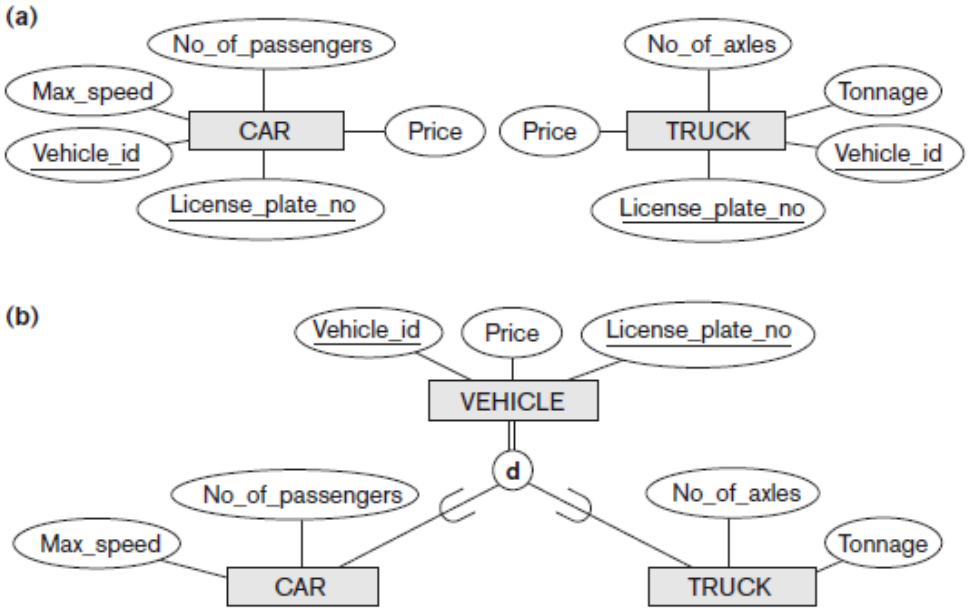
Example

Consider **STUDENT** and **FACULTY** entities.

- STUDENT (ID, Name, Course)
- FACULTY (ID, Name, Designation)

Common attributes are combined into a generalized entity **PERSON**.

Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.



MODUEL-3

5. a. Explain about relational model constraints.

Relational model constraints are rules that restrict the values stored in a database to maintain data accuracy, consistency, and integrity. These constraints ensure that only valid data is entered into the relational database.

1. Domain Constraint

A domain is the set of valid values that an attribute can take.

- It specifies the data type, range, and format of values.
- Values outside the defined domain are not allowed.

Example:

- Age must be an integer between 18 and 60

- Salary must be a positive number
- Gender ∈ {‘M’, ‘F’, ‘O’}

Ensures attribute values are meaningful.

2. Key Constraint

A key constraint ensures that each tuple (row) in a relation is uniquely identifiable.

- A super key uniquely identifies a tuple.
- A candidate key is a minimal super key.
- A primary key is one selected candidate key.

Example:

- Student_ID uniquely identifies each student in the STUDENT table.

Prevents duplicate records.

---

3. Entity Integrity Constraint

The entity integrity constraint states that:

- Primary key attributes cannot have NULL values.

Reason:

- A tuple must be uniquely identifiable.

Example:

- Student_ID in the STUDENT table cannot be NULL.

Ensures every record represents a real entity.

---

4. Referential Integrity Constraint

The referential integrity constraint maintains consistency between related relations.

- A foreign key value must:
    - Match a primary key value in the referenced table, or
    - Be NULL.

Example:

- Dept_ID in STUDENT must exist in the DEPARTMENT table.

5. Semantic Constraints (User-Defined Constraints)

These constraints represent business rules specific to an application.

- Not directly supported by the relational model.
- Implemented using CHECK, ASSERTIONS, or TRIGGERS.

Example:

- A student cannot register for more than 6 courses.
- Salary of an employee must be greater than that of an intern

b. **By refereeing the following database schema**
EMPLOYEE (Fname, Minit, Lname, SSN, Bdate, Address, Sex, Salary Sup-SSN, Dno)
DEPARTMENT (Dname, Dnumber, Mgr-SSN, Mgr-start-date)
Dept-Locations (Dnumber, Dlocation)
Project (Pname, Pnumber, Plocation, Dnum)
Works-on (Essn, Pno, Hours)
Dependent (Essn, Dependent-name, Sex, Bdate, Relationship)
Show the relational algebra expressions for the following queries.
i) Retrieve the name and address of all employees who work for the 'Research' department.

Step 1: Select Research department

$$D_1 \leftarrow \sigma_{Dname='Research'}(DEPARTMENT)$$

Step 2: Join with EMPLOYEE

$$E_1 \leftarrow EMPLOYEE \bowtie_{EMPLOYEE.Dno=D_1.Dnumber} D_1$$

Step 3: Project required attributes

$$\pi_{Fname,Minit,Lname,Address}(E_1)$$

ii) Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' either as a worker or as a manager of the department that controls the project.

Case 1: Smith works on the project

$$E_1 \leftarrow \sigma_{Lname='Smith'}(EMPLOYEE)$$

$$P_1 \leftarrow \pi_{Pno}(WORKS\_ON \bowtie_{Essn=SSN} E_1)$$

Case 2: Smith is manager of the department controlling the project

$$D_1 \leftarrow DEPARTMENT \bowtie_{Mgr\text{-}SSN=SSN} \sigma_{Lname='Smith'}(EMPLOYEE)$$

$$P_2 \leftarrow \pi_{Pnumber}(PROJECT \bowtie_{Dnum=Dnumber} D_1)$$

iii) List the names of managers who have at least one dependent.

Step 1: Get managers' SSN

$$M \leftarrow \pi_{Mgr\text{-}SSN}(DEPARTMENT)$$

Step 2: Find managers with dependents

$$MD \leftarrow M \bowtie_{Mgr\text{-}SSN=Essn} DEPENDENT$$

Step 3: Get manager names

$$\pi_{Fname,Minit,Lname}(EMPLOYEE \bowtie_{SSN=Mgr\text{-}SSN} MD)$$

iv) Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary.

Using **aggregate functions**:

$$\Upsilon_{SUM(Salary),\ MAX(Salary),\ MIN(Salary),\ AVG(Salary)}\ (EMPLOYEE)$$

v) For each project, retrieve the project number, the project name, and the number of employees who work on that project.

Step 1: Join PROJECT and WORKS_ON

$$PW \leftarrow PROJECT \bowtie_{Pnumber=Pno} WORKS\_ON$$

Step 2: Group and count employees

$$\Upsilon_{Pnumber,Pname;\ COUNT(Essn)}(PW)$$

Q.6 a. Explain how the basic operations deal with constraints violations with examples.

The basic database operations—INSERT, DELETE, and UPDATE—must ensure that relational model constraints are not violated. When a violation occurs, the DBMS either rejects the operation or takes corrective action.

INSERT Operation and Constraint Violations

Possible Violations

1. Domain Constraint
2. Key Constraint
3. Entity Integrity Constraint
4. Referential Integrity Constraint

Examples

(a) Domain Constraint Violation

```
INSERT INTO EMPLOYEE(SSN, Salary)
VALUES ('E101', -5000);
```

DELETE Operation and Constraint Violations

Possible Violation

- Referential Integrity Constraint

  DELETE FROM DEPARTMENT

  WHERE Dnumber = 5;

If employees belong to department 5, then deleting this tuple causes **dangling references.**

Possible DBMS Actions

1. **RESTRICT / NO ACTION** – Reject deletion
2. **CASCADE** – Delete related employee tuples
3. **SET NULL** – Set Dno in EMPLOYEE to NULL

UPDATE Operation and Constraint Violations

Possible Violations

1. Domain constraint
2. Key constraint
3. Entity integrity
4. Referential integrity

Operation	Possible Constraint Violations
INSERT	Domain, Key, Entity, Referential
DELETE	Referential
UPDATE	Domain, Key, Entity, Referential

b. Explain DIVISION operation with example.

- The division operation (÷) is used when we want to find tuples in one relation that are related to all tuples in another relation.
- It is usually applied to queries like "Find all X that are related to all Y".

If we have two relations:

- **R(A, B)**
- **S(B)**

The division **R ÷ S** gives a relation **T(A)** such that:

$$T = \{a \mid \text{for every } b \in S, (a, b) \in R\}$$

In other words: Return all a values from R that are associated with every b in S.

DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.

(a)

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

(b)

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

T

B
b1
b4

MODULE-4

Q.7 a. Explain INSERT, DELETE, UPDATE statements in SQL taking suitable examples

INSERT Statement

Adds new rows (tuples) into a table.

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

```
INSERT INTO EMPLOYEE (SSN, Fname, Lname, Dno, Salary)
VALUES ('E101', 'Alice', 'Smith', 5, 50000);
```

DELETE Statement

Removes existing rows from a table based on a condition.

```
DELETE FROM table_name
WHERE condition;
```

```
DELETE FROM EMPLOYEE
WHERE SSN = 'E101';
```

```
DELETE FROM EMPLOYEE;
```

UPDATE Statement-Modifies existing data in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
UPDATE EMPLOYEE
SET Salary = 55000
WHERE SSN = 'E102';

UPDATE EMPLOYEE
SET Salary = Salary * 1.1
WHERE Dno = 5;
```

b. What is Normalization? Explain 2NF, 3NF and with examples.

Definition:
Normalization is the process of organizing the attributes and relations of a database to reduce redundancy and avoid anomalies (insertion, deletion, update).
It divides large tables into smaller, well-structured tables while preserving data dependencies.

1. Eliminate redundant data.
2. Ensure data consistency.
3. Simplify data maintenance.

1NF

- A relation is in 1NF if all attributes are atomic (no repeating groups or arrays).
- Each column contains only single values.

StudentID	StudentName	Subjects
101	Alice	Math, Physics
102	Bob	Chemistry

Convert to 1NF:

StudentID	StudentName	Subject
101	Alice	Math
101	Alice	Physics
102	Bob	Chemistry

2NF

A relation is in 2NF if:

1. It is in **1NF**, and
2. **Every non-prime attribute is fully functionally dependent on the whole primary key** (no partial dependency).

Key Concept:

- Partial dependency occurs when a **non-key attribute depends only on part of a composite key**.

Example (1NF Table with Composite Key):

Example (1NF Table with Composite Key):

StudentID	CourseID	StudentName	CourseName
101	C1	Alice	Math
101	C2	Alice	Physics
102	C1	Bob	Math

- **Composite Key:** (StudentID, CourseID)
- **Partial Dependency:** StudentName depends only on StudentID.
- **Solution:** Split into two tables:

Student Table (2NF):

StudentID	StudentName
101	Alice
102	Bob

Course Table (2NF):

StudentID	CourseID	CourseName
101	C1	Math
101	C2	Physics
102	C1	Math

Third Normal Form (3NF)

Definition:

A relation is in 3NF if:

1. It is in **2NF**, and
2. There is **no transitive dependency** (non-key attribute depends on another non-key attribute).

Key Concept:

- Transitive dependency occurs when **A → B → C** (B is non-key).

Example (2NF Table):

EmpID	EmpName	DeptID	DeptName
E1	John	D1	Sales
E2	Alice	D2	HR

- **Transitive Dependency:** EmpID → DeptID → DeptName
- **Solution:** Split into two tables:

Employee Table (3NF):

EmpID	EmpName	DeptID
E1	John	D1
E2	Alice	D2

Department Table (3NF):

DeptID	DeptName
D1	Sales
D2	HR

Q.8 a. List and explain informal design guidelines for relation schemas.

1. **Clear semantics of attributes**
   Each relation should represent a single entity or relationship.
   All attributes must clearly describe that entity.
   This avoids confusion and improper data representation.

2. **Reduce redundant data storage**

The same data should not be stored repeatedly in multiple tuples.
Redundancy leads to update, insertion, and deletion anomalies.
Using separate relations and keys reduces duplication.

3. **Minimize NULL values**
   Relations should be designed to avoid unnecessary NULL values.
   Attributes not applicable to all tuples should be separated.
   This improves clarity and query efficiency.
4. **Avoid spurious tuples (ensure lossless decomposition)**
   Decomposition of relations must not produce incorrect tuples.
   Lossless join property should be satisfied.
   Natural joins should recreate the original relation correctly.

5. **Design relations that are easy to explain and understand**
   Relation schemas should be simple and meaningful.
   Attribute names must be clear and self-explanatory.
   Simple designs are easier to maintain and understand.
6. **Enforce integrity constraints (keys, domains, references)**
   Primary keys ensure uniqueness of tuples.
   Foreign keys maintain relationships between relations.
   Domain constraints restrict valid attribute values.

b. By refereeing the following database schema
Employee (Fname, Minit, Lname, SSN, Bdate, Address, Sex, Salary Sup-SSN, Dno)
Department (Dname, Dnumber, Mgr-SSN, Mgr-start-date)
Dept-Locations (Dnumber, Dlocation)
Project (Pname, Pnumber, Plocation, Dnum)
Works-on (Essn, Pno, Hours)
Dependent (Essn, Dependent-name, Sex, Bdate, Relationship)
Write queries in **SQL**.
i) Retrieve all the employee names who are working for department number 5.

```
SELECT Fname, Minit, Lname
FROM EMPLOYEE
WHERE Dno = 5;
```

ii) Retrieve all the projects which are controlled by department number 4

```
SELECT Pname
FROM PROJECT
WHERE Dnum = 4;
```

iii) Retrieve the names of employees who have no dependent.

```
SELECT Fname, Minit, Lname
FROM EMPLOYEE
WHERE SSN NOT IN (
```

```
    SELECT Essn
    FROM DEPENDENT
);
```

iv) Retrieve the employee name who is working on all the projects on which 'John Smith' works on.

```
SELECT E.Fname, E.Minit, E.Lname
FROM EMPLOYEE E
WHERE NOT EXISTS (
    (SELECT Pno
     FROM WORKS_ON W1, EMPLOYEE J
     WHERE J.Fname = 'John'
       AND J.Lname = 'Smith'
       AND W1.Essn = J.SSN)
    EXCEPT
    (SELECT W2.Pno
     FROM WORKS_ON W2
     WHERE W2.Essn = E.SSN)
);
```

v) Retrieve all the project numbers along with number of employees working on each project.

```
SELECT Pno, COUNT(Essn) AS Number_of_Employees
FROM WORKS_ON
GROUP BY Pno;
```

MODULE-5

Q.9 a. How are triggers and assertions defined in SQL? Explain.

A **trigger** is a set of SQL statements that are automatically executed (or "triggered") in response to certain events on a table or view, such as INSERT, UPDATE, or DELETE.

Key Points:

- Triggers are used to enforce **business rules**, **data integrity**, and **audit changes** automatically.
- They are defined to execute **before** or **after** the triggering event.
- A trigger can be **row-level** (executes for each affected row) or **statement-level** (executes once per SQL statement).

Syntax Example:

```
CREATE TRIGGER trg_before_insert_student
BEFORE INSERT ON STUDENT
FOR EACH ROW
BEGIN
  IF :NEW.age < 18 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Student must be at least 18 years old');
  END IF;
END;
```

Explanation:

- Trigger trg_before_insert_student fires **before** an insert on the STUDENT table.
- Checks if the age is less than 18 and prevents the insert if true.

---

2. Assertions in SQL

Definition:

An **assertion** is a constraint that specifies a **condition that must always be true** for the database. Unlike triggers, assertions **continuously enforce data integrity** across one or more tables.

Key Points:

- Assertions are used for **complex integrity constraints** that cannot be expressed by simple CHECK constraints.
- They are defined at the **schema level**, not table level.
- Not all RDBMS support SQL assertions (e.g., MySQL doesn't, Oracle uses triggers instead).

Syntax Example:

```
CREATE ASSERTION chk_student_course
CHECK (
  (SELECT COUNT(*) FROM ENROLLMENT WHERE grade = 'F') <= 5
);
```

- Ensures that the number of students failing a course does not exceed 5.
- The condition is always enforced, regardless of inserts or updates.

b. Describe the Two-Phase locking technique for concurrency control.

Two-Phase Locking (2PL) is a concurrency control protocol used in database systems to ensure serializability of transactions. It prevents conflicts that can arise when multiple transactions access the same data concurrently.

Key Concept:

- Each transaction goes through two distinct phases when acquiring and releasing locks:
    1. Growing Phase – The transaction may acquire locks but cannot release any.
    2. Shrinking Phase – The transaction may release locks but cannot acquire any new locks.

Steps / Phases of 2PL

1. **Growing Phase:**
    - Transaction requests locks on the required data items.
    - Can acquire **shared (S)** or **exclusive (X)** locks.
    - Cannot release any locks during this phase.
2. **Shrinking Phase:**
    - After releasing the first lock, the transaction enters the shrinking phase.
    - Can release locks one by one.
    - Cannot acquire any new locks after this phase begins.

Example Scenario

Consider two transactions:

- **T1:** Reads A, Writes B
- **T2:** Writes A, Reads B

2PL Execution:

Transaction	Action	Lock Acquired
T1	Read(A)	S-lock on A
T2	Write(A)	Wait (A locked by T1)
T1	Write(B)	X-lock on B
T1	Commit	Release locks A & B
T2	Write(A)	Acquires X-lock on A

This ensures **serializability** and prevents **lost updates** or **inconsistent reads**.

Advantages of 2PL

- Ensures **conflict-serializable schedules**.
- Simple and widely used in DBMS.

Disadvantages

- Can lead to **deadlocks** if two transactions wait for each other's locks.
- Can reduce system **concurrency**.

Variants of 2PL

1. **Strict 2PL:**
    - All locks are released only after the transaction commits.
    - Prevents cascading rollbacks.
2. **Rigorous 2PL:**
    - Even stricter than strict 2PL; all locks are held until the transaction completes.

Q.10 a. Explain views in SQL with example.

A view in SQL is a virtual table that is created using a SELECT query.
It does not store data physically; instead, it stores the query definition and displays data dynamically from one or more base tables.Views are used to simplify complex queries, provide security, and present customized data.

CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Example 1: Simple View

Create a view to show employees working in department 5.

CREATE VIEW Dept5_Employees AS

SELECT Fname, Lname, Salary

FROM EMPLOYEE

WHERE Dno = 5;

To retrieve data from the view:

SELECT * FROM Dept5_Employees;

- Updates are allowed **only if the view is simple**
- Views with **JOIN, GROUP BY, DISTINCT, or aggregate functions** are **not updatable**

```
UPDATE Dept5_Employees
SET Salary = Salary + 1000
WHERE Fname = 'Ravi';
```

Dropping a View

```
DROP VIEW Dept5_Employees;
```

Advantages of Views

1. **Security** – Restricts access to sensitive data
2. **Simplicity** – Hides complex queries
3. **Data Independence** – Changes in base tables do not affect users
4. **Customization** – Different views for different users
5.

b. Explain validation concurrency control techniques in databases.

Validation concurrency control is a technique used to maintain database consistency in a multi-user environment by assuming that conflicts are rare. Instead of locking data items, transactions are validated before commit to check whether conflicts occurred.It is also called Optimistic Concurrency Control (OCC).

- Transactions execute **without locks**
- Conflicts are checked **only at commit time**
- If a conflict is detected, the transaction is **rolled back**

This approach improves **system throughput** in low-conflict environments.

Each transaction has three phases:

1. Read Phase – Transaction reads data and updates local copies.
2. Validation Phase – DBMS checks for conflicts with other transactions to ensure serializability.
3. Write Phase – If validation succeeds, changes are written to the database; otherwise, the transaction is aborted and restarted.

It avoids deadlocks and provides high concurrency, making it suitable for read-intensive environments. However, it may cause frequent rollbacks in systems with many write conflicts.

Validation Conditions (Serializability Test)

Let:

- **RS(T)** = Read set of transaction T
- **WS(T)** = Write set of transaction T

Transaction **Ti** is valid if for every **Tj** such that:

$$finish(Tj) < validate(Ti)$$

One of the following holds:

1. WS(Tj) ∩ RS(Ti) = Ø
2. WS(Tj) ∩ WS(Ti) = Ø

If not, Ti is rolled back.

Example

Transactions

T1 reads and updates X

T2 reads X and commits before T1 validates

If **T2 writes X** and **T1 reads X**,

➡ Conflict occurs

➡ **T1 fails validation and is restarted**