DPG SCHOOL OF TECHNOLOGY & MANAGEMENT

(A unit of DPG Degree College, Sec-34, Gurugram)
(Affiliated to MDU Rohtak)
Recognized 2(f) by UGC & Accredited with 'A' Grade by NAAC

A
REVISION BOOK FOR
UPCOMING SEMESTER EXAM
WITH RESPECT TO **NEP CURRICULUM 2020** &
LATEST SYLLABUS PRESCRIBED BY
MDU , ROHTAK (HARYANA)

COURSE NAME : BACHELOR OF COMPUTER APPLICATION
PROGRAM NAME : DATABASE MANAGEMENT SYSTEM
PROGRAM CODE : 24BCA403DS03 CREDITS (L : T : P) : 3 : 0 : 1

SILENT FEATURES :

- DIRECT QUESTION FRAMED FROM THE SYLLABUS WITH EXAM ORIENTED SOLUTION .
- 20 SHORT QUESTION ANSWER FRAMED FROM EACH UNIT .

📘 Unit – 1: Database Management System

Topics covered: Introduction, applications, history, view of data, data abstraction, instances, schemas, DBMS environment, database languages, database models, ER model, entity, relationship, generalization, aggregation, conceptual design.

1. Define Database Management System (DBMS).
   A DBMS is software that manages data efficiently and allows users to perform operations like storing, retrieving, and updating data. It provides an interface between users and the database. It ensures data integrity, security, and reduces redundancy. Examples: MySQL, Oracle, MS Access.

2. What is a Database System?
   A database system consists of a collection of data and a set of programs to access and manage it. It provides a systematic way to store, retrieve, and manipulate information. It involves DBMS software, database, and users. Ensures data independence and consistency.

3. Define File Processing System.
   Earlier systems used file-based data storage. Each application maintained its own files. Problems included redundancy and inconsistency. No central control over data. DBMS replaced it to provide integration and security.

4. Define Data Abstraction.
   Data abstraction hides unnecessary details from users. It simplifies database interaction by showing only relevant levels.
   Levels:
   1. Physical Level – how data is stored.
   2. Logical Level – what data is stored.
   3. View Level – how data is viewed by users.

5. What are Instances and Schemas?

Instance: Actual content of the database at a particular time.

Schema: Overall logical structure or blueprint of a database. Schema rarely changes, while instances change frequently.

Example: Student table's structure = Schema ; records = Instances.

6. Define Data Independence.

   Ability to modify schema at one level without affecting the next higher level.

   Types :

   1. Logical Data Independence – change logical schema without altering external schema.
   2. Physical Data Independence – change physical storage without affecting logical schema.

7. Explain Database Languages.

   DBMS supports several languages for database operations:

   1. DDL (Data Definition Language) – defines schema.
   2. DML (Data Manipulation Language) – modifies data.
   3. DCL (Data Control Language) – manages access control.
   4. TCL (Transaction Control Language) – manages transactions.

8. What are the main components of a DBMS environment?

   Components :

   1. Hardware – physical devices for storage.
   2. Software – DBMS and OS.
   3. Data – stored information.
   4. Users – administrators and end-users.
   5. Procedures – rules for data handling.

9. Define Database Model.

   A database model defines how data is logically structured.

   Common models include :

   1. Hierarchical model.
   2. Network model.
   3. Relational model.

4. Object-oriented model.

The relational model is most widely used today.

10. Define Entity.

An entity represents a real-world object or concept about which data is stored. Example: Student, Employee, Product. Each entity has attributes describing its properties. Entities form the foundation of the ER model.

11. Define Attribute.

Attributes are characteristics or properties of an entity.

Example: Student ⟶ Roll No, Name, Age. Types: Simple, Composite, Derived, Multi-valued. Attributes are represented as ovals in an ER diagram.

12. What is a Relationship in DBMS ?

A relationship defines how entities are connected. Example: Student enrolled in Course.

Types :

1. One-to-One
2. One-to-Many
3. Many-to-Many

13. Define Entity-Relationship (ER) Model.

ER model represents data through entities and their relationships. Developed by Peter Chen (1976). Components: Entity, Attribute, Relationship, Cardinality. Used for conceptual database design before creating tables. 14. What is Generalization? Process of combining two or more entities with common attributes into a higher-level entity.

Example: Car and Truck ⟶ Vehicle.

Helps in reducing redundancy. Shown with an upward arrow in ER diagram.

15. Define Specialization.

Opposite of **generalization** . Divides a higher-level entity into sub-entities based on distinct features. Example: Employee ⟶ Permanent, Contractual. Used for detailed data

representation.

16. What is Aggregation in ER Model ?
    A concept used when relationships themselves have attributes. It treats a relationship as an entity for connecting with another entity.
    Example: "Teaches" between Teacher and Course linked with Department. Shown as a rectangle inside a diamond.

17. Explain Conceptual Database Design.
    Involves creating a high-level model of data using ER diagrams. Focuses on identifying entities, attributes, and relationships. It avoids physical details like indexing or storage. Provides a foundation for logical design.

18. Define Logical Database Design.
    Converts the conceptual model into a logical structure suitable for implementation.
    Example: converting ER model into relational tables. Includes defining primary and foreign keys. Ensures data consistency and integrity.

19. Define Physical Database Design.
    Specifies how data is physically stored and accessed. Includes file organization, indexing, and partitioning. Aims to optimize performance and storage efficiency. Based on hardware capabilities.

20. Write applications of Database Systems.
    Used in almost all fields of life:
    1. Banking – customer accounts, transactions.
    2. Education – student information system.
    3. Healthcare – patient records.
    4. E-commerce – order and inventory management.
    5. Government – citizen databases, ID systems.

📘 Unit – 2: Relational Model, Relational Algebra & Calculus, and SQL Queries

1. Define Relational Model.

   Proposed by E.F. Codd in 1970. Data is stored in tables called relations. Each table consists of rows (tuples) and columns (attributes). Provides mathematical foundation for data management. Uses keys and constraints to maintain data integrity.

2. Define Relation , Tuple , and Attribute.

   Relation : A table containing data.

   Tuple : A single row or record in a table.

   Attribute : A column representing a property.

   Example : Student (RollNo, Name, Age) ⟶ Relation name: Student ; Attributes: RollNo, Name, Age.

3. What is a Domain in the Relational Model ?

   A domain is the set of valid values an attribute can take. Ensures data integrity and consistency.

   Example: AGE domain ⟶ integers from 0–100.

   Each attribute is associated with exactly one domain.

4. What is a Relation Schema?

   A relation schema defines the structure of a relation. Includes the relation name and list of attributes.

   Example: STUDENT (RollNo, Name, Course, Marks).

   It acts as a blueprint for creating tables.

5. Define Primary Key.

   A Primary Key uniquely identifies each record in a table. It cannot have NULL or duplicate values.

   Example: RollNo in Student table.

   Ensures entity integrity and uniqueness.

6. Define Foreign Key.
   A Foreign Key links two tables together. It refers to the Primary Key of another table. Maintains referential integrity.
   Example: Dept_ID in Employee table references Dept_ID in Department table.

7. Define Candidate Key.
   Attributes that can uniquely identify tuples. Among them, one is chosen as the Primary Key.
   Example: (RollNo) and (AadharNo) both can be candidate keys. Ensures data uniqueness.

8. Define Super Key.
   A set of one or more attributes that uniquely identify a record. A Super Key may include extra attributes beyond those needed.
   Example: (RollNo, Name) ⟶ Super Key; RollNo alone ⟶ Candidate Key.

9. Define Alternate Key.
   Remaining candidate keys after selecting one as the primary key. They can also uniquely identify records.
   Example: AadharNo in Student table if RollNo is primary key.
   Provides alternative access paths.

10. What is Referential Integrity?
    Ensures that foreign key values always match primary key values. Prevents deletion of referenced records. Maintains consistency among related tables.
    Example: A student record cannot exist without a valid Course ID.

11. What is a View in SQL?

A view is a virtual table created from one or more tables. It doesn't store data itself.

SQL

CREATE VIEW view name AS SELECT * FROM table name;

Used for security and simplification .

12. What is Relational Algebra?

A procedural query language that operates on relations. Produces another relation as a result.

Basic operations : **Selection** (σ), **Projection** (π), **Union** (∪), **Difference** (—), **Cartesian Product** (×).

Foundation for SQL query processing.

13. Define Selection Operation (σ).

Selects rows (tuples) that satisfy a given condition.

Syntax :-

σ(condition)(Relation)

Example :-

σ(Age>18)(Student)

Selects students older than 18. Equivalent to SQL's WHERE clause.

14. Define Projection Operation (π).

Selects specific columns (attributes) from a relation.

Syntax :-

π(attribute list)(Relation)

Example :-

```
π(Name, Course)(Student)
```

Displays only Name and Course. Equivalent to SQL's SELECT clause.

15. Define Join Operation.

Combines related tuples from two or more tables.

Types : Inner Join, Outer Join, Natural Join.

Example :

```
SELECT * FROM Student NATURAL JOIN Course;
```

Used to retrieve data spread across multiple relations.

16. Define Union Operation.

Combines tuples from two relations with identical schemas. Duplicate rows are automatically removed.

```
Syntax: R ∪ S
```

Example: List of all students from two departments.

17. Define Intersection and Difference.

Intersection (R ∩ S): Returns common tuples between R and S.

Difference (R − S): Returns tuples in R not in S.

Both require relations with identical schemas.

18. What is Relational Calculus ?

A non-procedural query language. Specifies what data to retrieve, not how.

Two types:

1. Tuple Relational Calculus (TRC)
2. Domain Relational Calculus (DRC)

Used as a theoretical foundation for SQL.

19. What is a Query in SQL ?

A query is a command used to retrieve or manipulate data.

Example :-

```
sql SELECT Name FROM Student WHERE Marks > 80;
```

SQL queries can perform SELECT, INSERT, UPDATE, DELETE operations. It combines relational algebra concepts into practical syntax.

20. Define SQL and its Components.

Structured Query Language (SQL) is used to manage relational databases.

Divided into components :-

1. DDL: CREATE, ALTER, DROP.
2. DML: SELECT, INSERT, UPDATE, DELETE.
3. DCL: GRANT, REVOKE.
4. TCL: COMMIT, ROLLBACK.

▧ UNIT – 3: NORMALIZATION, TRANSACTION MANAGEMENT & CONCURRENCY CONTROL

1. Define Normalization.

Normalization is the process of organizing data to reduce redundancy and improve data integrity. It divides large tables into smaller ones and defines relationships between them. Improves data consistency and efficiency. Helps in maintaining database design clarity and prevents anomalies.

2. What is Redundancy ?

Redundancy means duplication of data in a database. It leads to wastage of memory and inconsistency in records. Example: Storing student address repeatedly in multiple tables. Normalization helps in eliminating redundancy effectively.

3. Define Decomposition.

   Decomposition is the process of breaking a complex table into smaller, simpler tables. It helps to remove redundancy and anomalies. It should maintain lossless join and dependency preservation.

   Example : Splitting a Student-Course table into separate Student and Course tables.

4. What are Anomalies in DBMS ?

   Anomalies occur when redundant data causes problems during updates.

   Types :

   1. Insertion Anomaly – Difficulty adding data.
   2. Deletion Anomaly – Loss of unintended data.
   3. Update Anomaly – Inconsistent updates.

   Normalization removes these anomalies.

5. Explain First Normal Form (1NF).

   A table is in 1NF if :

   1. Each cell holds atomic (single) values.
   2. Each record is unique.

   Example: No repeating groups or arrays.

   Ensures data is structured in simple rows and columns.

6. Explain Second Normal Form (2NF).

   A relation is in 2NF if it is in 1NF and all non-key attributes depend on the entire primary key. Removes partial dependency.

   Example : Composite key (RollNo, Course) $\longrightarrow$ Marks must depend on both keys, not one.

7. Explain Third Normal Form (3NF).

   A table is in 3NF if: 1. It is in 2NF. 2. It has no transitive dependency. Non-key attributes must depend only on primary key.

Example : Student(RollNo, DeptID, DeptName) ⟶ Move DeptName to a separate Department table.

8. What is BCNF (Boyce–Codd Normal Form) ?
   Stronger form of 3NF. A relation is in BCNF if every determinant is a candidate key. Removes anomalies not handled by 3NF.

   Example: Teacher(Subject, Department, Head) ⟶ Subject should determine Department uniquely.

9. Define Multivalued Dependency.
   Occurs when one attribute determines multiple independent attributes.
   Example: A student can have multiple hobbies and subjects.

   Represented as: X ⟶⟶ Y.
   Leads to data redundancy and is removed by 4NF.

10. Define Join Dependency.
    A condition where a table can be reconstructed by joining smaller tables. It ensures data preservation during decomposition. A relation free of join dependency is in 5NF. Helps maintain data consistency in complex databases.

11. Define Transaction in DBMS.
    A transaction is a single logical unit of work. It may involve multiple read/write operations.

    Example : Transfer ₹1000 from one account to another. A transaction must follow ACID properties for reliability.

12. Explain ACID Properties.

    - A – Atomicity: Transaction executes completely or not at all.

- C – Consistency: Maintains database correctness.
- I – Isolation: Concurrent transactions do not interfere.
- D – Durability: Changes persist even after system failure.

13. Define Transaction States.

States of a transaction :

1. Active – Execution started.
2. Partially Committed – All statements executed.
3. Committed – Changes saved permanently.
4. Failed – Error occurred.
5. Aborted – Rolled back to previous state.

14. Define Serializability.

Ensures that the result of concurrent transactions is the same as if executed serially.

Types :

1. Conflict Serializability.
2. View Serializability.

Prevents data inconsistency during concurrent execution.

15. Define Concurrency Control .

Technique to manage simultaneous execution of transactions. Prevents conflicts like lost updates, temporary inconsistency. Uses locks and timestamps. Ensures data integrity and isolation in multi-user environments .

16. What are Lock-Based Protocols ?

Concurrency control mechanism using locks to control access.

- Shared Lock (S): Read-only.
- Exclusive Lock (X): Read and write.

Prevents conflicts but may cause deadlocks .

17. What is a Deadlock ?

Occurs when two transactions wait indefinitely for each other's locked resources.
Example: T1 waits for A; T2 waits for B; both locked.
Solutions: Timeout, Deadlock Detection, and Prevention Techniques.

18. Define Lock Conversion .

Changing the mode of a lock during transaction execution.
Types :

1. Upgrade: Shared ⟶ Exclusive.
2. Downgrade: Exclusive ⟶ Shared .

Helps improve resource utilization in concurrent transactions.

19. What is Recoverability in Transactions ?

A schedule is recoverable if a transaction commits only after all transactions it depends on have committed. Prevents cascading rollbacks. Ensures database consistency in case of failure.

20. Define Concurrency without Locking.

Some systems avoid locks using timestamp ordering or optimistic concurrency control. Transactions execute freely but are validated before commit. Reduces deadlocks and improves performance.

UNIT – 4: CRASH RECOVERY, BACKUP & FILE ORGANIZATION

1. Define Database Recovery .

Database recovery is the process of restoring the database to a correct state after a failure. Ensures data consistency and integrity.

Types of recovery include : crash recovery, transaction recovery, and media recovery. Helps protect data from system errors or hardware failures.

2. Define Database Failure .

A failure is an event that causes a transaction or the whole system to stop functioning properly.

Types include :

1. Transaction failure.
2. System crash.
3. Media failure.

Database recovery techniques handle these to prevent data loss.

3. What is Crash Recovery ?

Restoring database state after a sudden system crash or power failure. Uses log files to redo or undo transactions. Ensures durability and consistency of committed data. Involves techniques like checkpoint and log-based recovery.

4. Define Storage Structure .

Physical arrangement of data on storage devices.

Includes :

1. Primary Storage (RAM).
2. Secondary Storage (Hard Disk).
3. Tertiary Storage (Backups).

Proper storage structure ensures quick data retrieval and durability.

5. Define Atomicity in Recovery .

Atomicity ensures that all parts of a transaction are executed completely or not at all. If failure occurs, partial changes are undone using rollback. It is one of the ACID properties of transactions.

6. Define Log-Based Recovery.

Each transaction maintains a log file of operations performed. Contains information like transaction ID, old value, and new value.
During recovery : Undo uncommitted transactions. Redo committed ones. Ensures durability and consistency.

7. What is a Checkpoint in DBMS ?
   A checkpoint is a saving point in the database log. It marks the state of the system at a specific time. Reduces recovery time by avoiding reprocessing all transactions.
   Example : Setting a checkpoint after every 10 transactions.

8. Define Log File.
   A log file records all database changes. Stored on stable storage separate from the main database. Contains transaction start, commit, and rollback entries. Used in recovery to redo or undo operations.

9. Define Media Failure.
   Occurs when a storage device (like hard disk) fails. Results in loss of part or all data stored on it. Recovery involves restoring the database from the latest backup.
   Example : Disk crash or magnetic failure.

10. Define Catastrophic Failure.
    A severe failure where both the database and log files are lost. Caused by hardware damage, fire, or corruption. Recovery possible only using offsite backups or remote copies. Requires complete database restoration.

11. Define Database Backup.
    A backup is a copy of database data stored for recovery. Protects against data loss due to system or media failure. Can be full, incremental, or differential backup.
    Example : Daily full backup + hourly incremental backup.

12. Explain Full Backup.

A complete copy of the entire database. Simplifies restoration since only one file is needed. Time-consuming and requires large storage. Usually performed once a week or daily for critical systems.

13. Define Incremental Backup.
    Backs up only data changed since the last backup. Reduces time and storage requirements. For recovery, both full and incremental backups are used together. Common in large databases with frequent updates.

14. Define Differential Backup.
    Copies all data modified since the last full backup. Restoration requires only the full backup and the latest differential backup. Balances between speed and simplicity.

15. Explain Remote Backup System.
    A remote backup is stored on a different physical location or cloud. Protects data from local disasters or hardware failures. Allows business continuity even after catastrophic failure.
    Example : Cloud backups on Google Drive or AWS.

16. Define File Organization.
    Refers to the method of storing records in files on disk.
    Common types : Sequential, Indexed, Hashed.
    Impacts data access speed and storage efficiency. The organization is chosen based on application requirements.

17. Explain Sequential File Organization.
    Records are stored in sequential order (usually by key value). Best for batch processing and range queries. Access is slow for random retrievals.
    Example : Payroll system processing.

18. Explain Indexed File Organization.

Uses an index to access records quickly. The index contains pointers to data locations. Improves search performance significantly.

Example : Library catalog system.

19. Explain Hashed File Organization.

Uses a hash function to compute the address of a record. Allows direct access to records. Efficient for point queries but not range queries. Example: Accessing customer by ID.

20. Explain Serial and Direct Files.

Serial Files : Records stored one after another without any order. Suitable for small datasets.

Direct Files : Use addresses to locate records instantly (similar to hashed files).

Serial ⟶ Simple but slow; Direct ⟶ Fast but complex.