CS3492 Database Management Systems

Purpose of Database System – Views of data – Data Models – Database System Architecture –Introduction to relational databases – Relational Model – Keys – Relational Algebra – SQL fundamentals– Advanced SQL features – Embedded SQL– Dynamic SQL

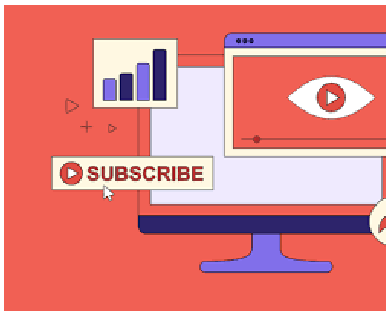
Important Notes

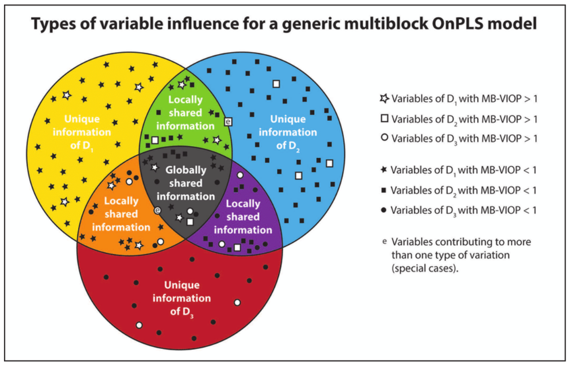
I. INTRODUCTION TO DATABASE SYSTEMS

Database systems manage data efficiently for various purposes.
Evolution from file-based systems to modern databases.
Data is a valuable organizational asset.

II. VIEWS OF DATA

Data abstraction simplifies complexity for different users.
End-user, logical, and physical views offer varying perspectives.

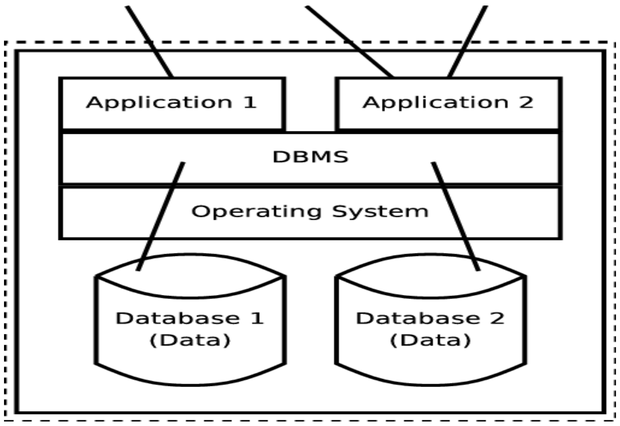



Types of variable influence for a generic multiblock OnPLS model
Unique information of D_1
Locally shared information
Unique information of D_2
Globally shared information
Locally shared information
Locally shared information
Unique information of D_3
Variables of D_1 with MB-VIOP > 1
Variables of D_2 with MB-VIOP > 1
Variables of D_3 with MB-VIOP > 1
Variables of D_1 with MB-VIOP < 1
Variables of D_2 with MB-VIOP < 1
Variables of D_3 with MB-VIOP < 1
e Variables contributing to more than one type of variation (special cases).

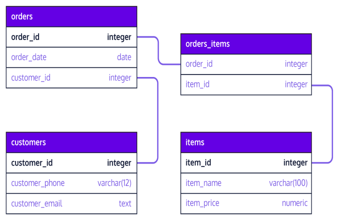
III. DATA MODELS

Data models define data's structure and relationships.
Types include hierarchical, network, relational, object-oriented, and entity-relationship models.
Choose the right data model based on requirements.

IV. DATABASE SYSTEM ARCHITECTURE



Components: hardware, software, data, users, procedures.
Key functions: storage, retrieval, transaction management, query processing, concurrency control, recovery, security.

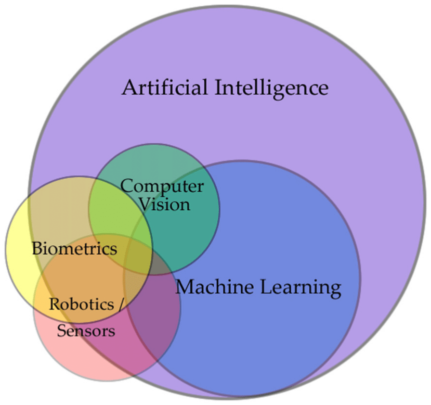


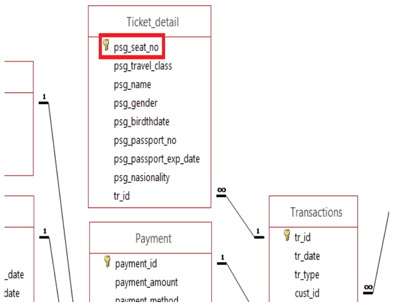
V. INTRODUCTION TO RELATIONAL DATABASES

Relational databases organize data in tables (relations).
Key characteristics: data integrity, normalization, SQL usage.
Advantages include data consistency and flexibility.

VI. RELATIONAL MODEL

Tables (relations) store data as rows (tuples) and columns (attributes).
Primary keys uniquely identify rows.
Foreign keys establish relationships between tables.
Integrity constraints ensure data accuracy.
Normalization reduces redundancy, denormalization optimizes for performance.

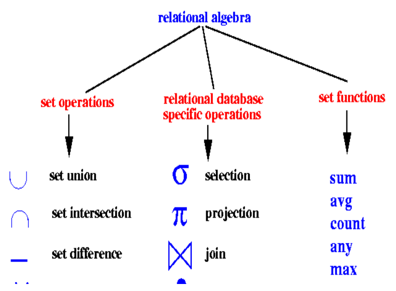




VII. KEYS IN RELATIONAL DATABASES

Primary Key: Uniquely identifies rows.
Candidate Key: Potential primary keys.
Super Key: Set of attributes that uniquely identifies rows.
Foreign Key: References primary keys in related tables.
Composite Key: Primary key with multiple attributes.
Surrogate Key: System-generated primary key.

VIII. RELATIONAL ALGEBRA



Basic Operators: Selection, Projection, Union, Intersection, Set Difference.
Join Operations: Inner, Outer, Self-Join.
Expressions combine operations.
Relational algebra underlies SQL.

IX. SQL FUNDAMENTALS

SQL is a standardized language for managing relational databases.
DML includes SELECT, INSERT, UPDATE, DELETE.
DDL includes CREATE TABLE, ALTER TABLE, DROP TABLE.
DCL manages access permissions (GRANT, REVOKE).

X. ADVANCED SQL FEATURES

Aggregate Functions: SUM, AVG, COUNT, MIN, MAX.
Subqueries: Nested queries within SQL statements.
Joins and Join Types: INNER, LEFT, RIGHT, FULL JOINs.
Transactions: COMMIT, ROLLBACK, SAVEPOINT.
Indexes improve query performance.
Views create virtual tables.

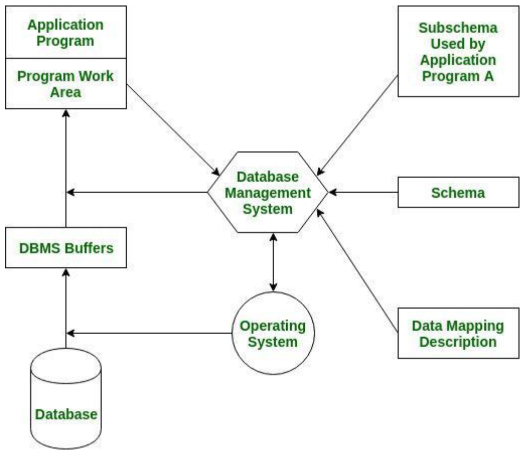
XI. EMBEDDED SQL

Embed SQL statements in application code.
Enhances database interactions in languages like C/C++, Java, Python.
Provides tight integration but may have security considerations.

XII. DYNAMIC SQL

Generate and execute SQL statements at runtime.
Useful for flexible database interactions.
Security concerns must be addressed.

<u>**Detailed Explanation:**</u>

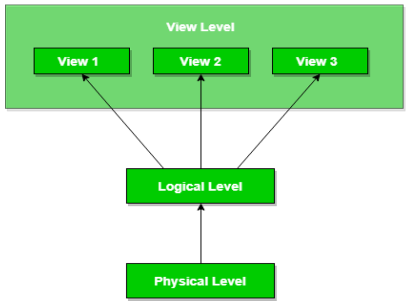


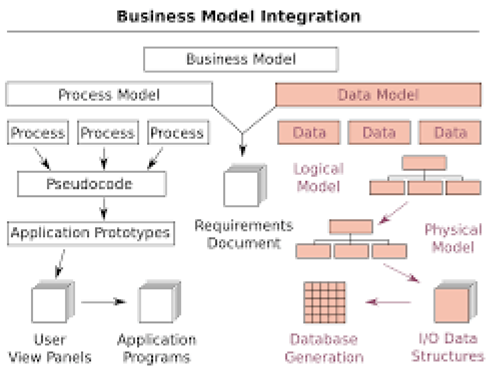
PURPOSE OF DATABASE SYSTEM

Data Storage: Databases are used to store large amounts of data in an organized and efficient way.
Data Retrieval: Databases allow users to easily retrieve data based on specific criteria.
Data Management: Databases provide a variety of tools and features for managing data, such as security, concurrency control, and backup and recovery.

VIEWS OF DATA

Physical View: The physical view shows how the data is actually stored on the computer disk.
Logical View: The logical view shows how the data is organized into tables, columns, and relationships.
User View: The user view shows how the data is presented to the user.



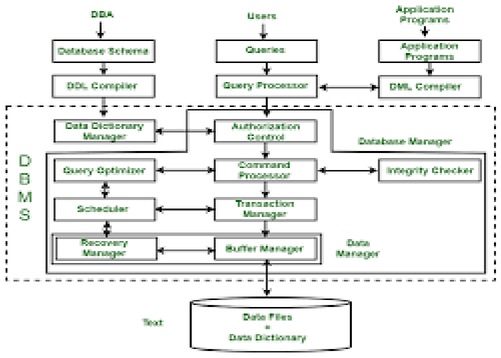


DATA MODELS

Hierarchical Data Model: A data model that organizes data in a tree-like structure.
Network Data Model: A data model that allows for more complex relationships between data items.
Relational Data Model: A data model that organizes data into tables, which are made up of rows and columns.

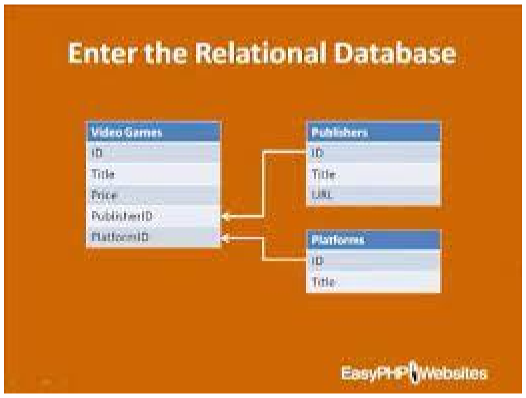
DATABASE SYSTEM ARCHITECTURE

Database: The collection of data that is stored and managed by the database system.Database Management System (DBMS): The software that



controls the database. It is responsible for creating, maintaining, and using the database.
Users: The people who interact with the database system. They can be database administrators, application developers, or end users.

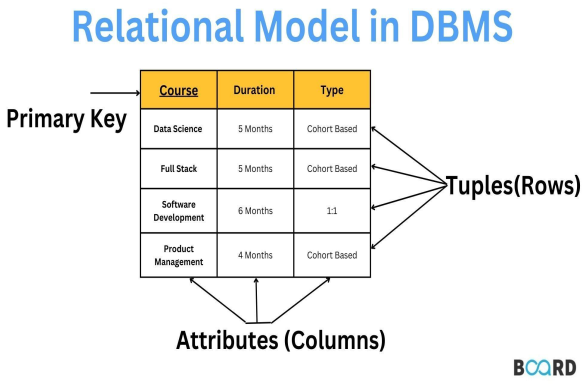
INTRODUCTION TO RELATIONAL DATABASES



Tables: A relational database is made up of tables. Each table has a set of columns, and each column has a set of rows.
Rows: A row in a table represents a single record.
Columns: A column in a table represents a single attribute of a record.

RELATIONAL MODEL

Entities: Entities are the real-world objects that the database is designed to represent. For example, a customer database might have entities such as customers, orders, and products.
Attributes: Attributes are the properties of entities. For example, a customer entity might have attributes such as name, address, and phone number.



Course	Duration	Type
Data Science	5 Months	Cohort Based
Full Stack	5 Months	Cohort Based
Software Development	6 Months	1:1
Product Management	4 Months	Cohort Based

Relationships: Relationships are used to represent the associations between entities. For example, a customer might place an order for a product.
KEYS

Primary Key: A primary key is a column or set of columns that uniquely identifies each record in a table.
Candidate Key: A candidate key is a column or set of columns that uniquely identifies each record in a table. A table can have multiple candidate keys, but only one can be the primary key.

Foreign Key: A foreign key is a column or set of columns in one table that references the primary key of another table. Foreign keys are used to create relationships between tables.

RELATIONAL ALGEBRA

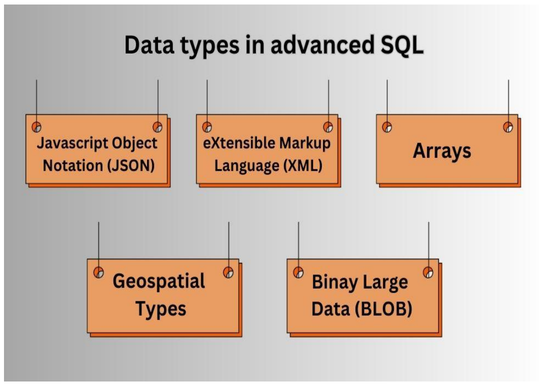
Relational operations: Relational algebra is a set of mathematical operations that can be used to manipulate relational databases. Relational algebra operations are used to write queries, which are requests for information from a database.

SQL FUNDAMENTALS

Queries: SQL (Structured Query Language) is the standard language for accessing and manipulating relational databases. Queries are used to retrieve data from a database, insert new data into a database, update existing data in a database, and delete data from a database.
CREATE TABLE: The CREATE TABLE statement is used to create a new table in a database.
INSERT: The INSERT statement is used to insert new data into a table in a database.
UPDATE: The UPDATE statement is used to update existing data in a table in a database.
DELETE: The DELETE statement is used to delete data from a table in a database.

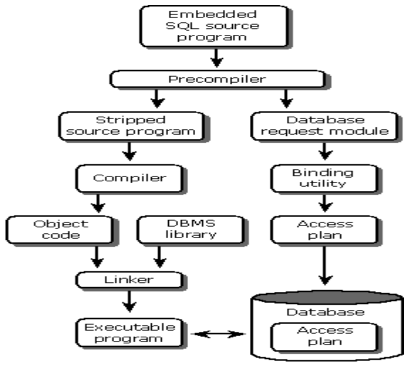
ADVANCED SQL FEATURES

Subqueries: Subqueries are queries that are nested within other queries.
Views: Views are logical tables that are based on one or more underlying tables.


Data types in advanced SQL
Javascript Object Notation (JSON)
eXtensible Markup Language (XML)
Arrays
Geospatial Types
Binay Large Data (BLOB)

Joins: Joins are used to combine data from multiple tables.
Triggers: Triggers are special procedures that are automatically executed when certain events occur, such as when a record is inserted, updated, or deleted.

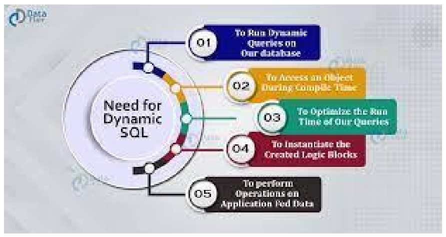
EMBEDDED SQL



SQL statements in programming languages: Embedded SQL is a way to embed SQL statements in programming languages such as C and Java. This allows developers to write database applications in their preferred programming language.

DYNAMIC SQL

SQL statements at runtime: Dynamic SQL is a way to generate SQL statements at runtime. This allows developers to write more flexible and dynamic database applications.



Need for Dynamic SQL
01 To Run Dynamic Queries on Our database
02 To Access an Object During Compile Time
03 To Optimize the Run Time of Our Queries
04 To Instantiate the Created Logic Blocks
05 To perform Operations on Application Fed Data

UNIT II DATABASE DESIGN

Entity-Relationship model – E-R Diagrams – Enhanced-ER Model – ER-to-Relational Mapping – Functional Dependencies – Non-loss Decomposition – First, Second, Third Normal Forms, Dependency Preservation – Boyce/Codd Normal Form – Multi-valued Dependencies and Fourth Normal Form – Join Dependencies and Fifth Normal Form

ENTITY-RELATIONSHIP MODEL (E-R MODEL)

Definition: A conceptual model used to represent entities, their attributes, and the relationships between them in a structured way.
Entities: Represent real-world objects or concepts.
Attributes: Properties or characteristics of entities.
Relationships: Connections between entities.

E-R DIAGRAMS (ENTITY-RELATIONSHIP DIAGRAMS)

Purpose: Visual representation of the E-R model using symbols like rectangles for entities, ovals for attributes, and diamond shapes for relationships.
Cardinality: Indicates how many instances of one entity are related to another (e.g., one-to-one, one-to-many, many-to-many).
Key Attributes: Attributes used to uniquely identify instances of an entity (e.g., primary keys).

ENHANCED-ER MODEL

Extensions: Enhancements to the basic E-R model, including more complex relationships, subtypes, and hierarchies.
Subtypes and Supertypes: Allows for entity specialization and generalization.
Aggregation: Combines multiple related entities into a single higher-level entity.

ER-TO-RELATIONAL MAPPING

Process: Mapping E-R diagrams to relational database schemas (tables).
Entities to Tables: Each entity becomes a table, with attributes as columns.
Relationships to Foreign Keys: Relationships become foreign keys in related tables.
Keys: Primary keys are defined based on the entity's key attributes.

FUNCTIONAL DEPENDENCIES

Definition: A property of a relation in which the value of one attribute uniquely determines the value of another attribute.
Candidate Keys: Attributes or combinations of attributes that uniquely identify tuples.
Full Functional Dependency: An attribute is functionally dependent on another if it's dependent on the entire candidate key.
Partial Functional Dependency: An attribute is functionally dependent on another if it's dependent on a part of the candidate key.

NON-LOSS DECOMPOSITION

Decomposition: Breaking a relation into multiple smaller relations.
Lossless Join: Ensuring that no information is lost during decomposition, and the original relation can be reconstructed from smaller relations.
Dependency Preservation: Preserving functional dependencies after decomposition.

NORMAL FORMS (FIRST, SECOND, THIRD)

First Normal Form (1NF): Ensures that attributes are atomic (indivisible) and each column has a single value.
Second Normal Form (2NF): Eliminates partial dependencies by moving them to separate tables.
Third Normal Form (3NF): Removes transitive dependencies by creating additional tables.

BOYCE/CODD NORMAL FORM (BCNF)

Definition: A stricter form of 3NF where every non-trivial functional dependency is on a superkey.
Superkey: A set of attributes that uniquely identifies tuples.

MULTI-VALUED DEPENDENCIES AND FOURTH NORMAL FORM (4NF)

Multi-valued Dependencies: When an attribute is multi-valued, meaning it can have multiple values for a single entity.
Fourth Normal Form (4NF): Addresses multi-valued dependencies, ensuring that they are represented properly in the database schema.
JOIN DEPENDENCIES AND FIFTH NORMAL FORM (5NF)

Join Dependencies: Express how a relation can be reconstructed by joining multiple smaller relations.
Fifth Normal Form (5NF): Ensures that join dependencies are satisfied and that there are no redundancy or loss issues when reconstructing relations.

Detailed explanation:

ENTITY-RELATIONSHIP MODEL (E-R MODEL)

The Entity-Relationship (E-R) model is a conceptual data model that is used to design relational databases. It is based on the idea of entities and relationships. Entities are real-world objects, such as customers, products, and orders. Relationships are the associations between entities, such as a customer placing an order for a product.

E-R DIAGRAMS

E-R diagrams are used to visually represent the E-R model. They consist of entities, relationships, and attributes. Entities are represented by boxes, relationships are represented by diamonds, and attributes are represented by ovals.

ENHANCED-ER MODEL

The Enhanced-ER model is an extension of the E-R model that includes additional features, such as generalization/specialization, aggregation, and association classes.

ER-TO-RELATIONAL MAPPING

ER-to-relational mapping is the process of converting an E-R model into a relational database schema. This process involves identifying the entities and relationships in the E-R model and then creating corresponding tables and foreign keys in the relational database schema.

FUNCTIONAL DEPENDENCIES

A functional dependency (FD) is a relationship between two sets of attributes in a table. An FD states that if two records have the same value for the first set of attributes, then they must also have the same value for the second set of attributes.

NON-LOSS DECOMPOSITION

Non-loss decomposition is the process of dividing a table into two or more tables without losing any information. This process can be used to normalize a table and improve its performance.

FIRST, SECOND, THIRD NORMAL FORMS

First, second, and third normal forms (1NF, 2NF, and 3NF) are three levels of normalization for relational databases. A table is in 1NF if and only if all of its attributes are atomic (i.e., they cannot be further divided into smaller parts). A table is in 2NF if and only if it is in 1NF and all of its non-key attributes are fully functionally dependent on the primary key. A table is in 3NF if and only if it is in 2NF and none of its non-key attributes are transitively functionally dependent on the primary key.

DEPENDENCY PRESERVATION

Dependency preservation is the process of ensuring that all of the functional dependencies in a database are preserved after a table is normalized.

BOYCE/CODD NORMAL FORM (BCNF)

Boyce/Codd normal form (BCNF) is a stricter normal form than 3NF. A table is in BCNF if and only if it is in 3NF and none of its non-key attributes are functionally dependent on any other non-key attribute.

MULTI-VALUED DEPENDENCIES AND FOURTH NORMAL FORM (4NF)

A multi-valued dependency (MVD) is a type of functional dependency that allows a single record to have multiple values for a single attribute. A table is in fourth normal form (4NF) if and only if it is in BCNF and none of its non-key attributes are multi-valued dependent on the primary key.

JOIN DEPENDENCIES AND FIFTH NORMAL FORM (5NF)

A join dependency (JD) is a type of functional dependency that arises when two tables are joined together. A table is in fifth normal form (5NF) if and only if it is in 4NF and none of its non-key attributes are join dependent on the primary key.

CONCLUSION

Database design is an important part of database development. By following the principles of database design, you can create databases that are efficient, reliable, and maintainable.

UNIT III TRANSACTIONS CS3492 Database Management Systems

Transaction Concepts – ACID Properties – Schedules – Serializability – Transaction support in SQL –
Need for Concurrency – Concurrency control –Two Phase Locking-Timestamp – Multiversion –
Validation and Snapshot isolation– Multiple Granularity locking – Deadlock Handling – Recovery
Concepts – Recovery based on deferred and immediate update – Shadow paging – ARIES Algorithm

IMPORTANT NOTES:

TRANSACTION CONCEPTS
Definition: A transaction is a sequence of one or more SQL operations treated as a single unit of work, ensuring consistency in a database.
Purpose: Transactions maintain the integrity of the database by guaranteeing that operations either all succeed or none at all.

ACID PROPERTIES (Atomicity, Consistency, Isolation, Durability)
Atomicity: Transactions are atomic, meaning they are treated as indivisible units; either all operations within a transaction are completed, or none are.
Consistency: Transactions bring the database from one consistent state to another, preserving integrity constraints.
Isolation: Transactions are isolated from each other to prevent interference and maintain data integrity.
Durability: Once a transaction is committed, its effects are permanent and will survive system failures.

SCHEDULES
Definition: A schedule is an arrangement of transactions' operations in time, showing the order in which they are executed.
Serial Schedule: Transactions execute one after the other.
Concurrent Schedule: Transactions execute in parallel.

SERIALIZABILITY
Definition: A schedule is serializable if it produces the same result as a serial schedule (i.e., the effect of concurrent execution is equivalent to some sequential execution).

TRANSACTION SUPPORT IN SQL
SQL provides statements such as BEGIN TRANSACTION, COMMIT, and ROLLBACK for managing transactions.
Transaction control commands allow you to explicitly start, commit, or roll back a transaction.

NEED FOR CONCURRENCY
Concurrency: Multiple transactions can execute concurrently, which improves system throughput and responsiveness.
Challenge: Managing concurrency to ensure data consistency and correctness.

CONCURRENCY CONTROL
Purpose: Ensures that transactions do not interfere with each other.
Two-Phase Locking: A protocol in which transactions acquire locks before accessing data and release them after completion.

TIMESTAMP
Timestamp-Based Concurrency Control: Assigns timestamps to transactions and data items to determine the order of access and resolve conflicts.

MULTIVERSION
Multiversion Concurrency Control: Allows transactions to see a snapshot of the database at the time the transaction started.

VALIDATION AND SNAPSHOT ISOLATION
Validation: Transactions are validated for conflicts before being committed.
Snapshot Isolation: Each transaction works with a snapshot of the database.

MULTIPLE GRANULARITY LOCKING
Fine-Grained Locking: Locking individual data items.
Coarse-Grained Locking: Locking larger portions of data, such as entire tables.

DEADLOCK HANDLING
Deadlock: Occurs when two or more transactions are waiting for each other to release locks.
Deadlock Detection: Identifying deadlocks and taking corrective action, such as aborting one of the involved transactions.

RECOVERY CONCEPTS
Purpose: Ensures database consistency after system failures.
Deferred Update: Updates are not made permanent until the transaction is committed.
Immediate Update: Updates are made permanent as soon as a write operation is executed.

SHADOW PAGING

Shadow Paging: A recovery technique where a shadow or duplicate copy of the database is maintained.

ARIES ALGORITHM

ARIES (Algorithm for Recovery and Isolation Exploiting Semantics): A widely used recovery algorithm that ensures durability and atomicity after a crash.

Detailed Explanations:

TRANSACTION CONCEPTS

A transaction is a unit of work that is performed on a database. It consists of a set of operations that are performed on the database and that must be completed successfully or rolled back completely.

ACID PROPERTIES

The ACID properties are a set of four properties that define the reliability and consistency of a transaction:

Atomicity: Atomicity ensures that a transaction is either completed successfully or rolled back completely.
Consistency: Consistency ensures that a transaction leaves the database in a consistent state.
Isolation: Isolation ensures that concurrent transactions do not interfere with each other.
Durability: Durability ensures that the changes made by a transaction are permanent, even if the system crashes.
Schedules

A schedule is a sequence of operations that are performed on a database by concurrent transactions.

SERIALIZABILITY
A schedule is serializable if it is equivalent to some serial schedule, where a serial schedule is a schedule in which transactions are executed one at a time.

TRANSACTION SUPPORT IN SQL
SQL provides a number of features for supporting transactions, including the following:

START TRANSACTION: Starts a new transaction.
COMMIT: Commits the current transaction and makes its changes permanent.
ROLLBACK: Rolls back the current transaction and undoes all of its changes.
Need for Concurrency

Concurrency is the ability of a database to support multiple transactions at the same time. Concurrency is important because it allows multiple users to access the database simultaneously and improves the performance of the database system.

CONCURRENCY CONTROL
Concurrency control is the process of ensuring that concurrent transactions do not interfere with each other. There are a number of different concurrency control techniques, including:

Two-phase locking: Two-phase locking is a concurrency control technique that uses locks to prevent concurrent transactions from interfering with each other.
Timestamp ordering: Timestamp ordering is a concurrency control technique that uses timestamps to order concurrent transactions.

Multiversion concurrency control: Multiversion concurrency control is a concurrency control technique that maintains multiple versions of data, which allows concurrent transactions to read different versions of the data without interfering with each other.
Validation and snapshot isolation: Validation and snapshot isolation is a concurrency control technique that uses a combination of validation and snapshot isolation to ensure that concurrent transactions are serializable.
Multiple granularity locking: Multiple granularity locking is a concurrency control technique that allows locks to be placed on different levels of granularity, such as rows, pages, and tables.

DEADLOCK HANDLING
A deadlock is a situation in which two or more transactions are waiting for each other to finish in order to proceed. There are a number of different deadlock handling techniques, including:

Deadlock detection: Deadlock detection identifies deadlocks when they occur.
Deadlock prevention: Deadlock prevention prevents deadlocks from occurring.
Deadlock recovery: Deadlock recovery recovers from deadlocks when they occur.

RECOVERY CONCEPTS
Recovery is the process of restoring the database to a consistent state after a system failure. There are two main types of recovery:

DEFERRED UPDATE: Deferred update recovery delays updating the database until the transaction commits.
Immediate update: Immediate update recovery updates the database immediately as the transaction executes.
Shadow paging

Shadow paging is a recovery technique that maintains a copy of the database, called the shadow page table, which is used to recover the database after a system failure.

ARIES ALGORITHM

The ARIES algorithm is a recovery algorithm that is used in many commercial database systems.

UNIT IV IMPLEMENTATION TECHNIQUES

RAID – File Organization – Organization of Records in Files – Data dictionary Storage – Column Oriented Storage– Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files –Static Hashing – Dynamic Hashing – Query Processing Overview – Algorithms for Selection, Sorting and join operations – Query optimization using Heuristics – Cost Estimation.

IMPORTANT NOTES:

RAID (Redundant Array of Independent Disks)
RAID is a technology that uses multiple hard drives to improve data redundancy, performance, or both.
RAID levels include 0, 1, 5, 10, and more, each with specific characteristics.

FILE ORGANIZATION
File: A collection of related records.
File Organization: The method used to arrange records within a file.
Common organizations include sequential, indexed, and hashed files.

ORGANIZATION OF RECORDS IN FILES
Records in a file can be organized using fixed-length or variable-length records.
Variable-length records are flexible but require additional metadata for storage.

DATA DICTIONARY STORAGE
A data dictionary stores metadata about the database, including information about tables, columns, indexes, and constraints.

COLUMN-ORIENTED STORAGE
In column-oriented storage, data is stored by columns rather than by rows, which can improve query performance for certain operations like aggregations.

INDEXING AND HASHING
Indexing and hashing techniques improve data retrieval efficiency.
Indexing: Creating an index structure to quickly locate data.
Hashing: Using a hash function to map data to storage locations.

ORDERED INDICES
Ordered Index: An index in which data is stored in sorted order, enabling efficient range queries.

B+ TREE INDEX FILES
A B+ tree is a self-balancing tree structure used in indexing, with efficient insertion, deletion, and range query capabilities.

B TREE INDEX FILES
A B tree is a balanced tree structure used for indexing, often used in databases for efficient data retrieval.

STATIC HASHING
Static hashing uses a fixed number of buckets, and records are assigned to buckets based on a hash function.

DYNAMIC HASHING
Dynamic hashing adjusts the number of buckets as data grows or shrinks, ensuring efficient distribution of data.

QUERY PROCESSING OVERVIEW
Query processing involves parsing, optimization, and execution of database queries.
It converts SQL queries into a plan for retrieving or modifying data.

ALGORITHMS FOR SELECTION, SORTING, AND JOIN OPERATIONS
Selection: Retrieving rows that meet specified conditions.
Sorting: Arranging data in a specific order.
Join: Combining data from multiple tables based on related columns.

QUERY OPTIMIZATION USING HEURISTICS
Query optimization aims to find the most efficient execution plan for a query.
Heuristic-based approaches use rules and estimates to choose a plan.

COST ESTIMATION
Cost estimation in query optimization involves estimating the resource usage (e.g., CPU, disk I/O) for different query execution plans.
The optimizer chooses the plan with the lowest estimated cost.

Detailed Explanations:

DISTRIBUTED DATABASES
Distributed databases are databases that are spread across multiple computers. Distributed databases can be used to improve performance, scalability, and reliability.

ARCHITECTURE

DISTRIBUTED DATABASES CAN BE CLASSIFIED INTO TWO MAIN TYPES:

Homogeneous distributed databases: Homogeneous distributed databases use the same database management system on all of the computers.
Heterogeneous distributed databases: Heterogeneous distributed databases use different database management systems on the different computers.
Data Storage

DISTRIBUTED DATABASES CAN STORE DATA IN A VARIETY OF WAYS, INCLUDING:

Fragmented data: Fragmented data is data that is divided into smaller pieces, called fragments. The fragments are stored on different computers.
Replicated data: Replicated data is data that is stored on multiple computers. This can be used to improve performance and reliability.
Transaction Processing

DISTRIBUTED DATABASES CAN PROCESS TRANSACTIONS IN A VARIETY OF WAYS, INCLUDING:

Two-phase commit: Two-phase commit is a transaction processing protocol that ensures that all of the computers involved in a transaction commit or rollback the transaction together.
Three-phase commit: Three-phase commit is a transaction processing protocol that is more complex than two-phase commit, but it can be used to improve performance and reliability.
Query Processing and Optimization

Query processing and optimization in distributed databases is more complex than in centralized databases. This is because the query optimizer needs to consider the location of the data and the network bandwidth when choosing the best way to execute a query.

NOSQL DATABASES

NOSQL databases are non-relational databases that are designed to be scalable and performant. NOSQL databases are often used for big data applications.

CAP THEOREM
The CAP theorem states that it is impossible for a distributed database to satisfy all of the following properties:

Consistency: All of the nodes in the database have the same data at all times.
Availability: The database is always available to users.
Partition tolerance: The database can continue to operate even if some of the nodes in the database are unavailable.
NOSQL databases typically sacrifice consistency in order to achieve availability and partition tolerance.

DOCUMENT BASED SYSTEMS
Document-based systems are NOSQL databases that store data in documents. Documents can be any type of data, such as JSON, XML, or BSON.

KEY-VALUE STORES
Key-value stores are NOSQL databases that store data in key-value pairs. Keys are typically strings, and values can be any type of data.

COLUMN BASED SYSTEMS
Column-based systems are NOSQL databases that store data in columns. This makes it efficient to perform analytical queries on the data.

GRAPH DATABASES
Graph databases are NOSQL databases that store data in graphs. A graph is a data structure that consists of nodes and edges. Nodes represent entities, and edges represent relationships between entities.

DATABASE SECURITY

Database security is the process of protecting databases from unauthorized access, use, disclosure, disruption, modification, or destruction.

SECURITY ISSUES

There are a number of security issues that can affect databases, including:

Unauthorized access: Unauthorized access is when someone who is not authorized to access a database gains access to it.
Data theft: Data theft is when someone steals data from a database.
Data corruption: Data corruption is when data in a database is modified or destroyed without authorization.
Denial of service: Denial of service attacks attempt to make a database unavailable to authorized users.
Access control based on privileges

Access control based on privileges (ACBP) is a database security model that grants users access to data based on their privileges. Privileges are permissions that allow users to perform certain operations on data.

ROLE BASED ACCESS CONTROL

Role-based access control (RBAC) is a database security model that grants users access to data based on their roles. Roles are groups of users that have the same privileges.

SQL INJECTION

SQL injection is a type of attack that injects SQL code into a database query. This can be used to gain unauthorized access to data or to modify or destroy data.

STATISTICAL DATABASE SECURITY

Statistical database security is a branch of database security that focuses on protecting the privacy of data in statistical databases. Statistical databases are databases that contain data that is used for statistical analysis.

FLOW CONTROL

Flow control is a database security technique that controls the flow of data into and out of

UNIT V ADVANCED TOPICS

Distributed Databases: Architecture, Data Storage, Transaction Processing, Query processing and optimization – NOSQL Databases: Introduction – CAP Theorem – Document Based systems – Key value Stores – Column Based Systems – Graph Databases. Database Security: Security issues – Access control based on privileges – Role Based access control – SQL Injection – Statistical Database security – Flow control – Encryption and Public Key infrastructures – Challenges

IMPORTANT NOTES:

DISTRIBUTED DATABASES

Architecture: Distributed databases span multiple locations or servers, often connected by a network. They can be centralized, decentralized, or hierarchical.

Data Storage: Data is distributed across multiple nodes or sites. Various replication and partitioning strategies are used.

Transaction Processing: Ensuring ACID properties in distributed environments can be challenging due to network latency and failures.

Query Processing and Optimization: Query processing involves distributed execution of queries across nodes. Optimization aims to minimize network traffic and latency.

NOSQL DATABASES (Not Only SQL)

Introduction: NoSQL databases are designed for flexibility, scalability, and handling unstructured or semi-structured data.

CAP Theorem (Consistency, Availability, Partition Tolerance): States that in a distributed system, you can achieve at most two out of the three

attributes (Consistency, Availability, and Partition Tolerance) at any given time.
Document-Based Systems: Store data in semi-structured documents (e.g., JSON, XML). Examples include MongoDB and CouchDB.
Key-Value Stores: Simplest NoSQL model, stores data in key-value pairs. Examples include Redis and Amazon DynamoDB.
Column-Based Systems: Organize data into columns rather than rows, suitable for analytical queries. Examples include Apache Cassandra and HBase.
Graph Databases: Designed for storing and querying data with complex relationships, often used in social networks and recommendation systems. Examples include Neo4j and Amazon Neptune.

DATABASE SECURITY
Security Issues: Concerns include unauthorized access, data breaches, and data integrity.
Access Control Based on Privileges: Assigning permissions and privileges to users and roles to control access to database objects.
Role-Based Access Control (RBAC): Assigning permissions based on user roles, simplifying security management.
SQL Injection: A security vulnerability where malicious SQL code is injected into input fields to manipulate a database.
Statistical Database Security: Protecting sensitive statistical data while allowing useful analysis.
Flow Control: Managing data flow to ensure only authorized users can access specific data.
Encryption and Public Key Infrastructures: Ensuring data confidentiality through encryption and PKI (Public Key Infrastructure).
Challenges: Security is an ongoing challenge due to evolving threats, new vulnerabilities, and complex access requirements.

Detailed Explanations:

DISTRIBUTED DATABASES
ARCHITECTURE: Distributed databases have a multi-node architecture that spans multiple locations or servers. They can be centralized, decentralized, or hierarchical. Centralized systems have a single central database server, while decentralized systems distribute data across multiple servers, each responsible for its data. Hierarchical systems combine centralized and decentralized models, with a central server and distributed servers at lower levels.

DATA STORAGE: In distributed databases, data is distributed across multiple nodes or sites. Various strategies are used for data storage, including data replication, data partitioning, and data fragmentation. Data replication involves maintaining copies of data on multiple servers to ensure availability and fault tolerance. Data partitioning divides data into partitions or shards, with each partition stored on a different server. Data fragmentation divides data into fragments, and each fragment can be stored on any available server.

TRANSACTION PROCESSING: Transaction processing in distributed databases aims to ensure ACID properties (Atomicity, Consistency, Isolation, Durability) despite challenges such as network latency and communication failures. Techniques like Two-Phase Commit (2PC) and Three-Phase Commit (3PC) are used to manage distributed transactions and ensure their consistency.

QUERY PROCESSING AND OPTIMIZATION: Query processing in distributed databases involves the execution of queries that span multiple nodes. Query optimization aims to minimize network traffic and latency.

It includes query distribution, where queries are divided into subqueries and sent to relevant nodes for execution, query coordination to decide which parts of a query can be executed locally, and data movement optimization to reduce the amount of data transferred between nodes for improved query performance.

NOSQL DATABASES (NOT ONLY SQL)

INTRODUCTION: NoSQL databases are designed to handle unstructured or semi-structured data and provide flexibility and scalability beyond traditional relational databases. They are categorized into various types, including document-based, key-value stores, column-based systems, and graph databases.

CAP THEOREM (CONSISTENCY, AVAILABILITY, PARTITION TOLERANCE): The CAP theorem states that in a distributed system, you can achieve at most two out of the three attributes (Consistency, Availability, and Partition Tolerance) at any given time. Consistency ensures that all nodes in the system have the same data view, availability guarantees that every request receives a response, and partition tolerance allows the system to continue operating even when network partitions occur.

DOCUMENT-BASED SYSTEMS: Document-based NoSQL databases store data in semi-structured documents, often in formats like JSON or XML. Each document can have different structures, allowing flexibility in data modeling. Examples include MongoDB and CouchDB.

KEY-VALUE STORES: Key-value stores are the simplest NoSQL model, storing data as key-value pairs. They are efficient for basic data retrieval and storage. Examples include Redis and Amazon DynamoDB.

COLUMN-BASED SYSTEMS: Column-based databases organize data into columns rather than rows, making them suitable for analytical queries. They are efficient for handling large volumes of data. Examples include Apache Cassandra and HBase.

GRAPH DATABASES: Graph databases are designed for storing and querying data with complex relationships. They are useful for applications like social networks and recommendation systems. Examples include Neo4j and Amazon Neptune.

DATABASE SECURITY

SECURITY ISSUES: Database security concerns encompass various aspects, including preventing unauthorized access, safeguarding against data breaches, and ensuring data integrity.

ACCESS CONTROL BASED ON PRIVILEGES: Access control mechanisms grant or deny permissions to users and roles based on their privileges. This controls who can perform specific actions on database objects like tables, views, and procedures.

ROLE-BASED ACCESS CONTROL (RBAC): RBAC simplifies access control by assigning permissions to roles and then assigning roles to users. This makes it easier to manage permissions in large systems.

SQL INJECTION: SQL injection is a security vulnerability where malicious SQL code is injected into input fields to manipulate a database. It can be prevented by using prepared statements and input validation.

STATISTICAL DATABASE SECURITY: Statistical databases must protect sensitive statistical data while allowing useful analysis. Techniques include adding noise to data and implementing differential privacy.

FLOW CONTROL: Flow control mechanisms restrict data flow and enforce security policies to ensure that only authorized users can access specific data.

Encryption and Public Key Infrastructures: Encryption protects data confidentiality. Public Key Infrastructures (PKIs) manage keys and digital certificates for secure communication.

CHALLENGES: Security is an ongoing challenge due to evolving threats, new vulnerabilities, and the need to balance security with usability and performance. Continuous monitoring and adaptation are essential to address security challenges effectively.