Database Management Systems Interview Preparation Guide

Basic Concepts

- **Database**: Organized collection of structured data
- **DBMS**: Software to manage databases; provides interface between data and users
- **Data Independence**: Ability to modify schema without affecting applications
  - **Logical Independence**: Change logical schema without changing applications
  - **Physical Independence**: Change physical storage without changing logical schema
- **Schema**: Database structure definition
  - **Physical Schema**: How data is stored physically
  - **Logical Schema**: Programmers' view of database
  - **View Schema**: End users' view of database

Data Models

- **Relational Model**: Data in tables (relations) with rows and columns
- **Entity-Relationship Model**: Entities, attributes, and relationships
- **Object-Oriented Model**: Data represented as objects
- **Hierarchical Model**: Tree-like structure (parent-child relationships)
- **Network Model**: Records with links between them (graph structure)

Relational Database Concepts

- **Relation**: Table with rows and columns
- **Tuple**: Row in a relation
- **Attribute**: Column in a relation
- **Degree**: Number of attributes in a relation
- **Cardinality**: Number of tuples in a relation
- **Primary Key**: Unique identifier for each tuple
- **Foreign Key**: References primary key of another relation
- **Candidate Key**: Attribute(s) that can uniquely identify a tuple
- **Super Key**: Set of attributes containing a candidate key
- **Composite Key**: Primary key made of multiple attributes

Normalization

- **Purpose**: Minimize redundancy and dependency; prevent anomalies

- **Normal Forms**:
  - **1NF**: Atomic values; no repeating groups
  - **2NF**: 1NF + no partial dependency of non-key attributes on the key
  - **3NF**: 2NF + no transitive dependency of non-key attributes on the key
  - **BCNF**: Every determinant is a candidate key
  - **4NF**: No multi-valued dependencies
  - **5NF**: No join dependencies

SQL (Structured Query Language)

- **DDL (Data Definition Language)**: CREATE, ALTER, DROP, TRUNCATE
- **DML (Data Manipulation Language)**: SELECT, INSERT, UPDATE, DELETE
- **DCL (Data Control Language)**: GRANT, REVOKE
- **TCL (Transaction Control Language)**: COMMIT, ROLLBACK, SAVEPOINT

Joins

- **INNER JOIN**: Returns rows when matches exist in both tables
- **LEFT JOIN**: Returns all rows from left table and matching rows from right
- **RIGHT JOIN**: Returns all rows from right table and matching rows from left
- **FULL JOIN**: Returns rows when matches exist in either table
- **CROSS JOIN**: Cartesian product of two tables
- **SELF JOIN**: Joining a table to itself

Indexes

- **Purpose**: Speed up data retrieval
- **Types**:
  - **Primary Index**: On primary key
  - **Secondary Index**: On non-primary fields
  - **Clustered Index**: Determines physical order of data
  - **Non-Clustered Index**: Logical order with pointers to physical data
- **Data Structures**: B-Tree, B+ Tree, Hash Index, Bitmap Index

Transactions

- **ACID Properties**:
  - **Atomicity**: All operations complete or none do
  - **Consistency**: Database moves from one valid state to another

    - **Isolation**: Concurrent transactions don't affect each other
    - **Durability**: Committed changes are permanent
- **Transaction States**: Active, Partially Committed, Committed, Failed, Aborted
- **Concurrency Control**: Ensures isolation between transactions
- **Locking Mechanisms**: Shared (read) locks, Exclusive (write) locks

Concurrency Control

- **Problems**:
    - **Dirty Read**: Reading uncommitted data
    - **Non-Repeatable Read**: Re-reading returns different values
    - **Phantom Read**: Re-execution of query finds new rows
    - **Lost Update**: Two transactions overwrite each other
- **Solutions**:
    - **Lock-Based**: Two-phase locking (2PL)
    - **Timestamp-Based**: Assign timestamps to transactions
    - **Multiversion Concurrency Control (MVCC)**: Multiple versions of data
- **Isolation Levels**:
    - **Read Uncommitted**: Lowest isolation; allows dirty reads
    - **Read Committed**: Prevents dirty reads
    - **Repeatable Read**: Prevents non-repeatable reads
    - **Serializable**: Highest isolation; prevents phantom reads

Deadlocks

- **Definition**: Circular waiting condition where each transaction holds resources needed by others
- **Necessary Conditions** (all four must occur simultaneously):
    - **Mutual Exclusion**: At least one resource must be held in non-sharable mode
    - **Hold and Wait**: Process holding resources can request additional resources
    - **No Preemption**: Resources cannot be forcibly taken from processes
    - **Circular Wait**: Circular chain of processes, each waiting for resource held by next
- **Detection**:
    - **Wait-For Graph**: Directed graph showing which transaction waits for which
    - **Timeout Detection**: Assuming deadlock if transaction waits too long
- **Prevention**:
    - **Requiring all locks at once**: Eliminates hold and wait

  - **Resource ordering**: Acquire locks in predefined order to prevent cycles
  - **Wait-Die/Wound-Wait**: Based on transaction timestamps
- **Resolution**:
  - **Victim Selection**: Choose transaction to abort (based on age, progress, resources held)
  - **Rollback**: Complete vs partial transaction rollback
  - **Starvation Prevention**: Ensuring no transaction is repeatedly victimized

Database Recovery

- **Types of Failures**:
  - **Transaction Failure**: Logical errors, system errors, deadlocks
  - **System Failure**: Power outage, hardware/software failures
  - **Media Failure**: Disk crash, storage corruption
  - **Network Failure**: Communication breakdown between distributed components
- **Recovery Techniques**:
  - **Log-Based Recovery**: Write-ahead logging (WAL)
    - Undo logging: Records old values before updates
    - Redo logging: Records new values of updates
    - Undo/Redo logging: Records both old and new values
  - **Shadow Paging**: Maintains shadow copy of database
  - **Remote Backup Systems**: Standby database for disaster recovery
- **Recovery Operations**:
  - **Undo**: Reverses uncommitted transactions (rollback)
  - **Redo**: Reapplies committed transactions to restore state
  - **ARIES**: Algorithm for Recovery and Isolation Exploiting Semantics
    - Log sequence numbers (LSNs) for ordering
    - Write-ahead logging protocol
    - Repeating history during redo
    - Logging compensation records during undo
- **Checkpoint Mechanisms**:
  - **Simple Checkpoint**: Pauses all transaction processing
  - **Fuzzy Checkpoint**: Allows transactions during checkpoint
  - **Incremental Checkpoint**: Checkpoints subsets of database
- **Recovery Management**:
  - **Recovery Manager**: Handles database recovery process

  - **Buffer Manager**: Manages cached data pages
  - **Transaction Manager**: Tracks transaction states
- **High Availability Solutions**:
  - **Replication**: Maintaining multiple copies of data
  - **Failover Systems**: Automatic switching to standby system
  - **Data Mirroring**: Real-time duplication of data
  - **Clustering**: Multiple servers acting as one system

Database Security

- **Authentication**: Verifying user identity
- **Authorization**: Access control through privileges
- **Encryption**: Data encryption at rest and in transit
- **Auditing**: Tracking database access and changes
- **SQL Injection**: Attack using malicious SQL code

Advanced Topics

Distributed Databases

- **Fragmentation**: Horizontal (rows) vs Vertical (columns)
- **Replication**: Copying data to multiple locations
- **Consistency Models**: Strong, Eventual consistency
- **CAP Theorem**: Consistency, Availability, Partition tolerance (pick two)

NoSQL Databases

- **Types**:
  - **Key-Value**: Redis, DynamoDB
  - **Document**: MongoDB, CouchDB
  - **Column-Family**: Cassandra, HBase
  - **Graph**: Neo4j, OrientDB
- **BASE Properties**: Basically Available, Soft state, Eventually consistent
- **When to use**: High volume, flexible schema, horizontal scaling

Data Warehousing

- **OLTP vs OLAP**: Transactional vs Analytical processing
- **Star Schema**: Fact table surrounded by dimension tables

- **Snowflake Schema**: Normalized dimension tables
- **ETL Process**: Extract, Transform, Load
- **Data Mining**: Pattern discovery in large datasets

Common Interview Questions

Basic Concepts

1. **Explain DBMS advantages?** Data sharing, redundancy control, data consistency, security, integrity constraints.
2. **What are database anomalies?** Insertion, deletion, and update anomalies caused by data redundancy.
3. **Entity vs Attribute vs Relationship?** Entity is object, attribute is property, relationship is association between entities.

SQL and Database Design

1. **Difference between DELETE, DROP, and TRUNCATE?** DELETE removes rows (can be rolled back), DROP removes tables (can't be rolled back), TRUNCATE removes all rows quickly (can't be rolled back).
2. **Write SQL query for nth highest salary?** Using subquery: `SELECT * FROM Employee e1 WHERE n-1 = (SELECT COUNT(DISTINCT salary) FROM Employee e2 WHERE e2.salary > e1.salary)`.
3. **What is normalization and why use it?** Process to reduce redundancy and dependency; prevents anomalies.

Transactions and Deadlocks

1. **Explain ACID properties?** Atomicity (all or nothing), Consistency (valid state transitions), Isolation (concurrent transactions don't interfere), Durability (persistent after commit).
2. **What is a deadlock and how to prevent it?** Circular wait for resources; prevent by resource ordering or requiring all locks at once.
3. **Phantom read vs dirty read?** Phantom: new rows appear; Dirty: reading uncommitted data.

Indexing and Performance

1. **How does indexing improve performance?** Creates data structure for faster lookups without scanning entire table.
2. **Clustered vs non-clustered index?** Clustered: defines physical order; only one per table. Non-clustered: logical ordering; multiple allowed.
3. **When would you avoid indexes?** Small tables, frequent updates/inserts, columns with low selectivity.

Database Recovery

1. **Explain the importance of write-ahead logging?** Ensures data durability by writing log records before actual data changes.
2. **How does ARIES recovery algorithm work?** Uses logging with LSNs, repeats history during redo, and uses compensation logs during undo.
3. **What's the difference between warm and cold backup?** Warm backup is taken while database is running; cold backup requires shutdown.

Advanced Topics

1. **CAP theorem explanation?** Distributed systems can't simultaneously guarantee Consistency, Availability, and Partition tolerance.
2. **When to use NoSQL over RDBMS?** High volume data, flexible schema, horizontal scaling, eventual consistency acceptable.
3. **OLTP vs OLAP?** OLTP: transactional processing (many short transactions); OLAP: analytical processing (complex queries on historical data).

Optimization Techniques

- **Query Optimization**: Rewriting queries for efficiency
- **Denormalization**: Adding redundancy for performance
- **Partitioning**: Horizontal (sharding) vs Vertical
- **Caching**: Storing frequently accessed data in memory
- **Connection Pooling**: Reusing database connections
- **Proper Indexing**: Creating right indexes on right columns
- **Avoid SELECT ***: Specify only needed columns
- **Execution Plan Analysis**: Understanding query performance