DATABASE MANAGEMENT SYSTEM (DBMS) – IN-DEPTH EXPLANATION

UNIT I: INTRODUCTION TO DATABASES

1. Data and Information

Data are raw, unorganized facts (numbers, text, symbols) without context. **Information** is processed, organized data that has meaning and context.

Real-time Example: In a hospital:

- **Data**: "98.6", "101", "John", "37.5"
- **Information**: "Patient John Smith has a fever of 101°F" (processed data with context)

2. Database

A structured collection of logically related data stored electronically to serve multiple applications.

Real-time Example: Amazon's product database containing all item details, prices, inventory, customer reviews, and seller information in organized tables.

3. Database Management System (DBMS)

Software that manages databases, providing an interface between users/applications and the database.

Real-time Example: MySQL managing Uber's database - handling rider accounts, driver locations, trip history, and payments.

4. File System vs DBMS

File System: Each application has its own private files (data redundancy, inconsistency).
DBMS: Centralized data shared by multiple applications (minimal redundancy, maintained consistency).

Real-time Example:

- **File System**: Bank using separate files for loans, accounts, and customers (customer address might differ between files)
- **DBMS**: Bank using Oracle DBMS where all departments access the same customer address data

5. Characteristics of DBMS

- **Self-describing**: Contains catalog (metadata) describing structure
- **Program-data independence**: Changes to data don't require program changes
- **Multiple views**: Different users see different data presentations
- **Concurrent access**: Multiple users can access simultaneously

Real-time Example: Google's Spanner DBMS handles millions of concurrent users accessing Gmail, YouTube, and Google Drive with consistent views.

6. Advantages and Disadvantages of DBMS

Advantages: Data sharing, consistency control, security, backup/recovery, standards enforcement. **Disadvantages**: High initial cost, complexity, performance overhead, vulnerability.

Real-time Example:

- **Advantage**: Airline reservation system ensures when a seat is booked, all agents see it instantly
- **Disadvantage**: When AWS RDS has an outage, all dependent applications fail

7. Database Users

- **Naive Users**: End-users (ATM customers, online shoppers)
- **Application Programmers**: Develop applications using DBMS

- **Sophisticated Users**: Analysts, data scientists querying directly
- **Database Administrators (DBA)**: Manage, secure, backup database

Real-time Example:

- **Naive User**: Customer browsing Netflix
- **DBA**: Netflix engineer managing Cassandra clusters

8. Database System Architecture

1-Tier: Direct usage on personal computer (Microsoft Access on local machine) **2-Tier**: Client-server (Desktop application connecting to database server) **3-Tier**: Web applications (Browser → Web Server → Database Server)

Real-time Example:

- **3-Tier**: Online banking: Browser → Apache Server → Oracle Database

9. Database Applications

Banking, airlines, universities, sales, manufacturing, human resources.

Real-time Example:

- **Sales**: Shopify's database managing millions of e-commerce stores
- **Manufacturing**: SAP HANA managing supply chain, inventory, production

UNIT II: DATA MODELS AND DATABASE DESIGN

1. Data Models

Conceptual representation of how data is organized.

Hierarchical Model: Tree structure (parent-child) *Real-time Example*: Early IBM's IMS for NASA Apollo program organizing spacecraft components

Network Model: Graph structure (owner-member) *Real-time Example*: IDS (Integrated Data Store) for complex manufacturing processes

Relational Model: Tables (rows/columns) *Real-time Example*: PostgreSQL managing Instagram's user data, posts, comments

Object-Oriented Model: Objects with attributes/methods *Real-time Example*: MongoDB storing complex JSON documents for real-time analytics

2. Entity-Relationship (ER) Model

Conceptual design using entities, attributes, relationships.

Real-time Example: Designing university database with entities: Student, Course, Professor

3. Entities, Attributes, and Relationships

- **Entity**: Distinct object (Student, Product)
- **Attribute**: Property of entity (Student_ID, Product_Price)
- **Relationship**: Association between entities (Student "enrolls in" Course)

Real-time Example:

- **Entity**: "Employee" at Google
- **Attribute**: "Employee_ID", "Salary", "Department"
- **Relationship**: "Manages" between Manager and Team

4. Types of Attributes

- **Simple**: Atomic (Phone_Number)
- **Composite**: Can be divided (Name → First, Last)
- **Derived**: Calculated (Age from Birth_Date)
- **Multivalued**: Multiple values (Phone_Numbers)

Real-time Example:

- **Composite**: Amazon customer "Address" → Street, City, ZIP
- **Derived**: "Account_Age" calculated from "Join_Date"

5. Cardinality and Participation Constraints

Cardinality: 1:1, 1:N, M:N **Participation**: Total (double line) or Partial (single line)

Real-time Example:

- **1:1**: One passport belongs to one person
- **M:N**: Students enroll in multiple courses, courses have multiple students
- **Total**: Every bank account must belong to a customer

6. ER Diagrams

Visual representation using Chen/IE notation.

Real-time Example:

- **Hospital ERD**: Patient, Doctor, Appointment entities with relationships
- **Rectangle**: Entity (Patient)
- **Diamond**: Relationship (Treats)
- **Oval**: Attribute (Patient_ID)

7. Mapping ER Model to Relational Model

Converting ER design to database tables.

Real-time Example:

- **Entity "Student"** → STUDENT table

- **Relationship "Enrolls"** → ENROLLMENT table with foreign keys
- **M:N relationship** → Junction table

8. Database Design Process

Requirements → Conceptual Design → Logical Design → Physical Design

Real-time Example: Designing Twitter database:

1. **Requirements**: Users, tweets, followers, likes
2. **Conceptual**: ER diagram
3. **Logical**: Relational schema
4. **Physical**: MySQL tables with indexes

UNIT III: RELATIONAL MODEL AND SQL

1. Relational Model Concepts

- **Relation**: Table
- **Tuple**: Row
- **Attribute**: Column
- **Domain**: Set of permissible values

Real-time Example:

- **Relation**: "ORDERS" table in Amazon
- **Tuple**: One specific order (#12345)
- **Attribute**: "Order_Date", "Amount"
- **Domain**: "Order_Status" ∈ {Pending, Shipped, Delivered}

2. Relational Schema and Instance

Schema: Structure (CREATE TABLE statement) **Instance**: Actual data at a moment

Real-time Example:

- **Schema**: "CREATE TABLE Employees (...)"
- **Instance**: Actual employee records on January 1, 2024

3. Keys

Super Key: Any set of attributes uniquely identifying tuples **Candidate Key**: Minimal super key **Primary Key**: Chosen candidate key **Foreign Key**: References primary key of another table

Real-time Example:

- **Candidate Keys**: (Email) or (SSN) for Employee
- **Primary Key**: Employee_ID
- **Foreign Key**: Dept_ID in Employee table references Department table

4. Integrity Constraints

- **Entity**: Primary key not null, unique
- **Referential**: Foreign key must reference existing primary key
- **Domain**: Values within allowed set

Real-time Example:

- **Referential**: Cannot delete customer with existing orders
- **Domain**: "Age" cannot be negative

5. Structured Query Language (SQL)

DDL (Data Definition): CREATE, ALTER, DROP *Real-time Example*: `CREATE TABLE Users (id INT PRIMARY KEY, name VARCHAR(50));`

DML (Data Manipulation): SELECT, INSERT, UPDATE, DELETE *Real-time Example*: `UPDATE Products SET price = 19.99 WHERE id = 101;`

DCL (Data Control): GRANT, REVOKE *Real-time Example*: `GRANT SELECT ON Customers TO sales_team;`

TCL (Transaction Control): COMMIT, ROLLBACK *Real-time Example*: Banking transaction - debit one account, credit another, then COMMIT

6. SQL Queries

SELECT: `SELECT name, salary FROM Employees WHERE dept = 'Sales';` *Real-time Example*: Amazon querying products under $50 in Electronics

JOINS:

- **INNER**: Matching rows only
- **LEFT**: All from left, matching from right

Real-time Example:

```
-- Get all orders with customer info
SELECT Orders.id, Customers.name
FROM Orders
```

```
INNER JOIN Customers ON Orders.customer_id = Customers.id;
```

Subqueries: Query within query *Real-time Example*:

```
-- Products more expensive than average
SELECT name, price FROM Products
WHERE price > (SELECT AVG(price) FROM Products);
```

Aggregate Functions: COUNT, SUM, AVG, MAX, MIN *Real-time Example*:

```
-- Total sales by region
SELECT region, SUM(sales) FROM Orders
GROUP BY region
HAVING SUM(sales) > 100000;
```

UNIT IV: NORMALIZATION AND TRANSACTION MANAGEMENT

1. Database Anomalies

Insertion: Cannot add data without other data **Deletion**: Deleting data removes unrelated information **Update**: Changing data requires multiple updates

Real-time Example: Unnormalized table: Student_Course(Student_ID, Name, Course1, Course2, Course3)

- **Insertion**: Cannot add Course4 without redesign
- **Deletion**: Deleting a course might remove student info
- **Update**: Changing student name requires updating multiple rows

2. Functional Dependencies

X → Y means X determines Y

Real-time Example:

- **Email → Name**: If same email, same name
- **Order_ID → Order_Date**: Order ID determines order date

3. Normalization

1NF: Atomic values, no repeating groups *Real-time Example*: Instead of "Courses: Math,Science", create separate rows

2NF: 1NF + no partial dependencies *Real-time Example*: Order table with (Order_ID, Product_ID, Product_Name) violates 2NF (Product_Name depends on Product_ID, not full key)

3NF: 2NF + no transitive dependencies *Real-time Example*: Employee(Emp_ID, Dept, Dept_Location) violates 3NF (Dept_Location depends on Dept, not Emp_ID)

BCNF: Stronger 3NF where every determinant is candidate key

4. Transactions

Logical unit of work

Real-time Example: Bank transfer:

```
BEGIN TRANSACTION;
UPDATE Accounts SET balance = balance - 100 WHERE account = 'A';
UPDATE Accounts SET balance = balance + 100 WHERE account = 'B';
COMMIT;
```

5. ACID Properties

- **Atomicity**: All or nothing
- **Consistency**: Valid state before/after
- **Isolation**: Concurrent transactions don't interfere
- **Durability**: Committed changes persist

Real-time Example:

- **Atomicity**: Either both debit and credit happen, or neither
- **Isolation**: While transferring $100 from A to B, another transaction sees either old or new balance, not intermediate

6. Concurrency Control

Managing simultaneous transactions

Real-time Example: Ticket booking system:

- User A and B viewing same seat
- Lock prevents both from booking it

7. Serializability

Schedule equivalent to serial execution

Real-time Example: Bank transactions interleaved but result same as sequential execution

8. Deadlock

Two transactions waiting for each other

Real-time Example:

- T1 locks Account A, wants Account B
- T2 locks Account B, wants Account A
- Both wait forever

9. Recovery Management

Restoring database after failure

Real-time Example: Using transaction logs to redo committed transactions after system crash

UNIT V: STORAGE, INDEXING, AND SECURITY

1. Database Storage Structure

- **Memory hierarchy**: Cache → RAM → SSD → HDD → Tape
- **Storage manager**: Buffer management

Real-time Example:

- **Hot data**: Frequently accessed user profiles in RAM (Redis)
- **Cold data**: Old logs on HDD

2. File Organization

- **Heap**: Unordered
- **Sequential**: Ordered
- **Hash**: Hash function determines location

Real-time Example:

- **Sequential**: Student records sorted by ID for quick range queries
- **Hash**: User sessions by session ID for O(1) access

3. Indexing

Primary Index: On ordered key field *Real-time Example*: Index on Student_ID (primary key)

Secondary Index: On non-key field *Real-time Example*: Index on Student_Email for fast lookups

B-Tree: Balanced tree supporting range queries *Real-time Example*: Indexing product prices for "price BETWEEN 10 AND 50"

B+ Tree: Only leaves store data, better for range queries *Real-time Example*: Most relational databases (MySQL, PostgreSQL) use B+ trees

4. Query Processing

Steps: Parsing → Optimization → Execution

Real-time Example:

```
SELECT * FROM Orders
WHERE customer_id = 123 AND order_date > '2024-01-01';
```

1. Parse SQL
2. Choose index on customer_id
3. Filter by date
4. Return results

5. Query Optimization

Choosing most efficient execution plan

Real-time Example: Query with JOIN and WHERE:

- Option 1: Filter first, then join
- Option 2: Join first, then filter
- Optimizer chooses based on statistics

6. Database Security

Protecting from unauthorized access

Real-time Example:

- **Encryption**: Credit cards encrypted in database
- **Firewalls**: Allow only app servers to connect to database

7. Authorization and Authentication

Authentication: Verifying identity **Authorization**: Determining privileges

Real-time Example:

- **Authentication**: Username/password, biometrics
- **Authorization**: HR staff can see salaries, others cannot

8. Backup and Recovery

- **Full backup**: Complete database copy
- **Incremental**: Changes since last backup
- **Point-in-time recovery**: Restore to specific moment

Real-time Example:

- **Bank**: Nightly full backup + transaction log backups every 15 minutes
- **Accidental deletion**: Restore from backup

9. Distributed Databases (Introduction)

Data stored across multiple locations

Real-time Example:

- **Global company**: Customer data in regional databases (US, EU, Asia)
- **Benefits**: Faster local access, redundancy
- **Challenges**: Consistency across locations