NRCM
your roots to success...

Unit-1

Introduction to Database and System Architecture

Database Management Systems · Computer Science

Introduction to Database and System Architecture

Database Fundamentals, Design Principles & ER Modeling

Course: Database Management Systems

Date: May 2026

SEMESTER 2025-2026

Contents Overview

1

Introduction to Database and System Architecture

Database systems, data models, users, and structure

2

Introduction to Database Design

ER diagrams, relationships, and logical design

This presentation covers the fundamentals of **database systems** and their architecture, followed by an in-depth exploration of **database design** using the Entity-Relationship model. We will examine data models, system structure, ER diagrams, and the transition from conceptual to logical design.

Key Topics Covered:

- Database Systems and Applications
- Database vs File System
- View of Data & Data Models
- Database Users & Administrator
- Database System Structure
- ER Diagrams
- Relationships
- Extended ER Features
- Conceptual & Logical Design

PART 01

Database Fundamentals and System Architecture

Introduction to Database and System Architecture

Understanding database systems, their structure, and how they differ from traditional file systems

** This chapter covers database applications, file system comparison, data views, models, users, and system structure*

Database Systems and Their Applications

A **database** is a collection of related data organized for efficient retrieval and management. Database systems power critical applications across every industry.

Banking & Finance

Transaction processing, account management, fraud detection, and real-time balance updates

Requires ACID compliance and high concurrency

Healthcare

Electronic health records, patient management, medical history tracking, and prescription systems

HIPAA compliance and data security critical

E-Commerce

Inventory management, order processing, customer recommendations, and payment handling

High scalability and real-time analytics

Other Key Applications

Education (student records, LMS) · Government (tax, census) · Telecom (billing, CRM) · Airlines (reservations, scheduling) · Social Media (content, user graphs)

Database vs File System

File System

Traditional Data Storage Approach

- Data stored in **individual files**
- **Redundancy** and inconsistency
- Data **isolation** across files
- **Integrity** problems (no constraints)
- **Atomicity** issues on failure
- Concurrent access **anomalies**
- **Security** limitations
- Each app manages its own data

Database System

DBMS-Managed Data Storage

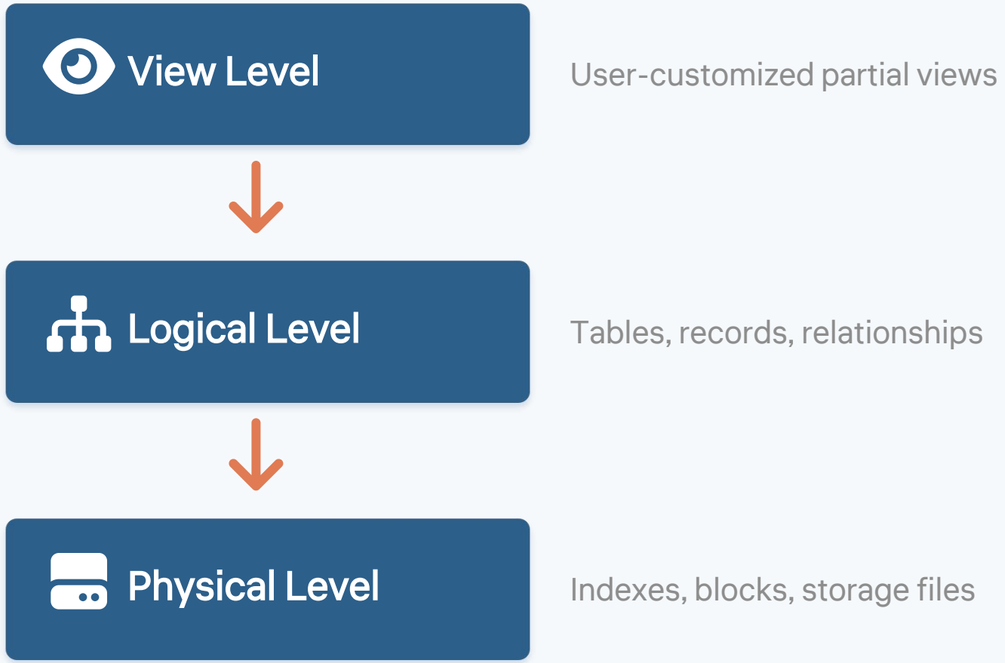
- **Centralized** data management
- Reduced redundancy (**normalization**)
- Data **consistency** enforced
- Data **independence**
- Concurrent **access control**
- Robust **security** mechanisms
- **Backup & recovery**
- Apps share data via DBMS

View of Data: Three Levels of Abstraction

Data abstraction hides complexity while allowing efficient storage and access. Each level provides a different perspective.

- **Physical Level**: How data is stored — lowest level with complex structures
- **Logical Level**: What data is stored and relationships among data
- **View Level**: Highest level showing only part of the database

Key Insight: Physical data independence allows changing storage without affecting logical schema



Schema and Instances

Schema: The logical structure of the database (like a blueprint).

Changes infrequently.

Instance: The actual content of the database at a point in time.

Changes constantly.

Analogous to **variables** (schema = type, instance = value)

Data Models

A data model is a collection of conceptual tools for describing data, relationships, semantics, and constraints.

Relational Model

Data organized in **tables** (relations) with rows and columns. Most widely used model. Based on mathematical set theory. Examples: MySQL, PostgreSQL, Oracle.

Entity-Relationship Model

Uses **entities, attributes, and relationships**. Primarily used for database design. Provides graphical representation via ER diagrams.

Object-Oriented Model

Data stored as **objects** with attributes and methods. Supports encapsulation, inheritance, and polymorphism. Examples: db4o, ObjectDB.

Semi-Structured Model

Flexible schema using **XML, JSON**. Used in NoSQL databases. Ideal for hierarchical and nested data. Examples: MongoDB, Cassandra.

Legacy models: Hierarchical (tree structure, IBM IMS) and Network (graph structure, CODASYL) — largely superseded by relational and NoSQL models

Database Users and Administrator

Types of Database Users

Naive Users

Interact through applications without knowing DBMS details (e.g., ATM users)

Application Programmers

Write code using DML calls embedded in host languages

Sophisticated Users

Interact directly with DBMS using query languages (SQL)

Specialized Users

Write specialized database applications (CAD, GIS, multimedia)

Database Administrator (DBA)

Schema Management

Define and modify database schema, storage structures, and access methods

Security & Authorization

Grant permissions, manage user accounts, enforce security policies

Routine Maintenance

Backups, performance tuning, integrity checks, and recovery procedures

Liaison Role

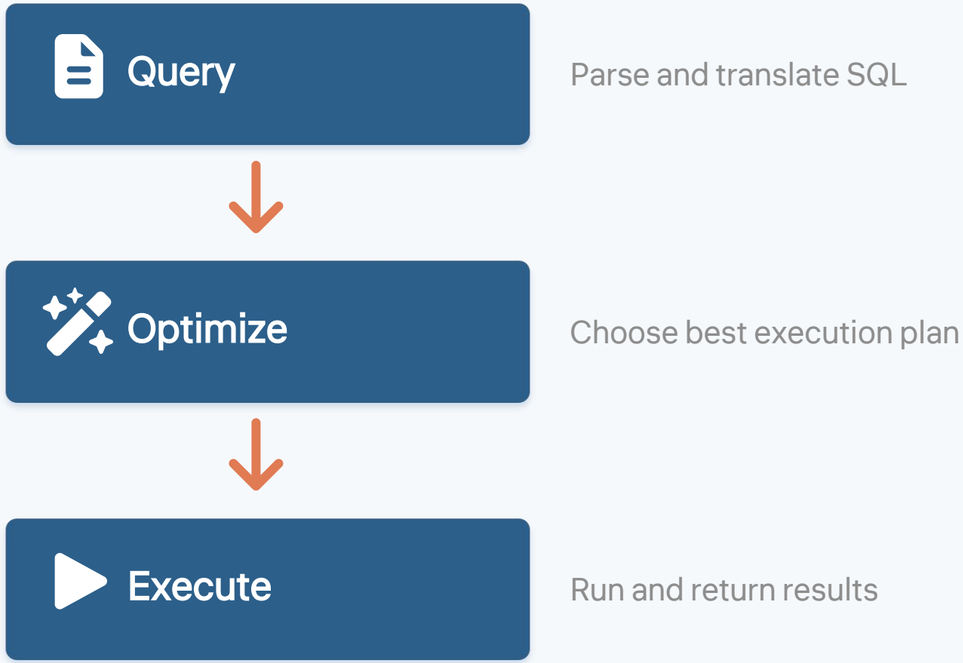
Bridge between users and technical staff; ensure data availability

Database System Structure

The DBMS acts as an intermediary between users and the physical database, with two major subsystems.

- **Storage Manager**: Interface to low-level data storage
- **Query Processor**: Parses, optimizes, executes queries
- **Transaction Manager**: Ensures ACID properties
- **Architecture Types**: 1-tier, 2-tier, 3-tier

ACID: Atomicity, Consistency, Isolation, Durability — guarantees reliable transaction processing



Storage Manager Components

- **Transaction Manager** — concurrency control
- **Buffer Manager** — disk-to-memory transfer
- **File Manager** — disk space allocation
- **Authorization & Integrity** — security

PART 02

Entity-Relationship Modeling and Database Design

Introduction to Database Design

Entity-Relationship modeling, conceptual design, and logical database design principles

** This chapter covers ER diagrams, entities, relationships, extended features, and logical design mapping*

ER Diagrams: Entities, Attributes & Entity Sets

ER diagrams use **rectangles** for entities, **ovals** for attributes, and **diamonds** for relationships.

Entity & Entity Set

An **entity** is a real-world object (e.g., Student, Course). An **entity set** is a collection of entities of the same type (e.g., all students in a university).

Simple vs Composite

Simple: Indivisible (e.g., age, salary). **Composite**: Can be divided into subparts (e.g., name = first_name + last_name, address = street + city + zip).

Single-valued vs Multi-valued

Single-valued: One value per entity (e.g., date_of_birth). **Multi-valued**: Multiple values possible (e.g., phone_numbers, email_addresses).

Stored vs Derived

Stored: Directly saved (e.g., date_of_birth). **Derived**: Computed from other attributes (e.g., age derived from date_of_birth). Also includes **null** attributes.

Relationships and Relationship Sets

Cardinality Ratios

How entities relate to each other

One-to-One (1:1)

One entity relates to exactly one other entity (e.g., Employee ↔ ParkingSpot)

One-to-Many (1:N)

One entity relates to multiple entities (e.g., Department → Employees)

Many-to-One (N:1)

Multiple entities relate to one entity (e.g., Employees → Department)

Many-to-Many (M:N)

Multiple entities relate to multiple (e.g., Students ↔ Courses)

Participation & Degrees

Constraints and relationship types

Total Participation

Every entity must participate in the relationship (double line in ER diagram)

Partial Participation

Entity participation is optional (single line in ER diagram)

Degree of Relationship

- Unary (recursive): One entity set
- Binary: Two entity sets
- Ternary: Three entity sets

Extended ER Features

Extended ER features provide greater modeling power for complex real-world scenarios.

Specialization

Top-down approach: identifying subsets of an entity set with distinct characteristics. Example: Person → Student, Employee (with unique attributes for each).

Generalization

Bottom-up approach: combining multiple entity sets into a higher-level entity. Example: Student + Employee → Person (common attributes factored out).

Inheritance Constraints

Disjoint: Subclasses cannot share entities. **Overlapping**: Subclasses can share entities. **Total**: Every entity must belong to a subclass. **Partial**: Optional.

Aggregation

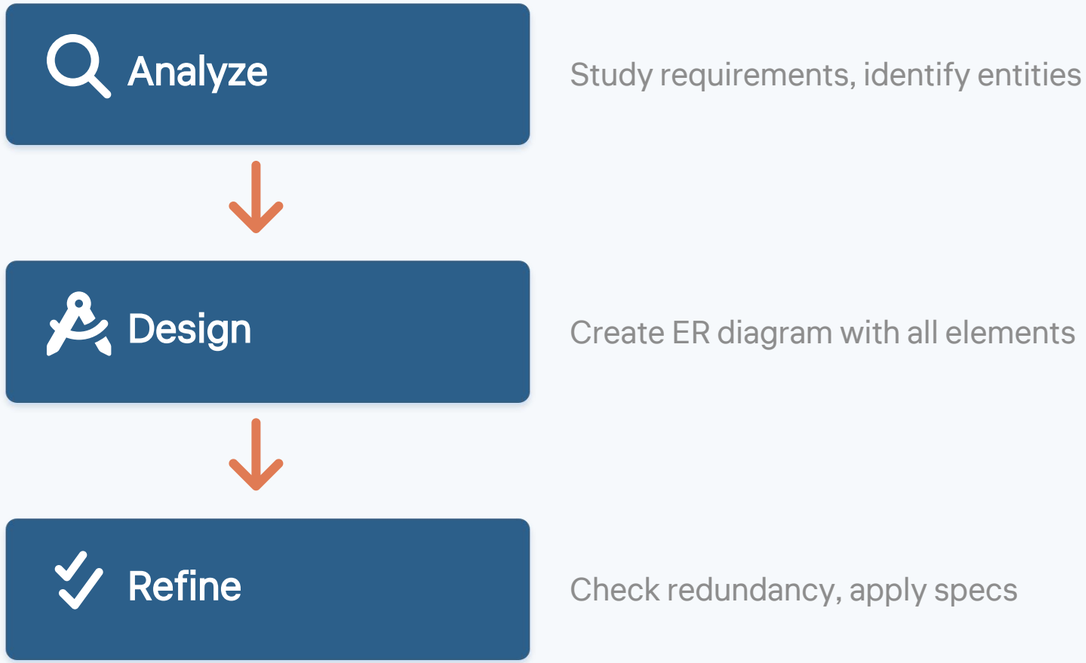
Treating a **relationship set as a higher-level entity**, allowing relationships with relationships. Example: "sponsorship" between Employee and a (Project-Department) relationship.

Conceptual Design with the ER Model

Translating requirements into a **conceptual schema** independent of any specific DBMS.

- **Step 1**: Identify entity sets from requirements
- **Step 2**: Identify relationships and cardinalities
- **Step 3**: Identify attributes and value sets
- **Step 4**: Determine primary keys

Goal: Capture all requirements while minimizing redundancy and avoiding design anomalies



Design Pitfalls to Avoid

- **Entity vs Attribute:** Use entity when it has its own properties
- **Entity vs Relationship:** Use relationship for associations
- **Prefer binary** over n-ary relationships when possible

ER to Relational Mapping

Converting conceptual schema to tables

- Each **entity set** → a table (relation)
- Each **attribute** → a column
- **Primary keys** are preserved
- **1:1** → FK in either table
- **1:N** → FK on "many" side
- **M:N** → Junction table
- **Weak entities** → FK of owner
- **Multi-valued** → Separate table

Normalization

Organizing data to minimize redundancy

1NF

All attributes atomic (indivisible values only)

2NF

No partial dependency (non-key attrs depend on full PK)

3NF

No transitive dependency (attrs depend only on PK)

BCNF

Every determinant is a candidate key (stricter than 3NF)

Thank You

Introduction to Database and System Architecture

Database Management Systems

Questions & Discussion

Database Systems · Data Models · ER Diagrams · Conceptual Design · Logical Design · Normalization