Database fundamentals summary (Part 1)

Agenda

1. **Introduction to Databases**
    - Before databases & Problems
2. **Main Components**
    - Database, DBMS, Application Program
3. **Advantages of DBMS**
4. **DBMS Architecture**
    - External, Conceptual, Physical Schemas
    - Data Models
5. **Entity Relationship Diagram (ERD)**
    - Purpose & Design Steps
    - Entities, Attributes, Relationships
6. **Types of Relationships**
    - Binary, Unary, Ternary
7. **Cardinality Ratios**
    - 1:1, 1:N, M:N, Recursive, Ternary
8. **Participation in Relationships**
    - Mandatory vs Optional
    - Attributes on relationships
9. **Logical Design (Relational Model)**
    - Tables, Primary Keys
    - Mapping rules

10. **Relationship Mappings Comparison**
    - 1:N, Unary, 1:1, M:N, Weak Entity, Ternary

Introduction

- **Before databases**: Organizations used **separate file systems**
- **Problems with separate file system**:
    - Data duplication
    - Data isolation (hard to access across departments)
    - Difficult data sharing
- **Database solution**: Collect related data in one central place.
- **DBMS (Database Management System):** software that creates and maintains the database.
- **Database System** = Data + DBMS software

Main Components of a Database System

1. **Database** → stores the actual data.
2. **DBMS** → software that controls and maintains the database.
    - One part processes queries from applications.
    - Another part retrieves or modifies data in the database.
3. **Application Program** → interface for the end user (e.g., e-commerce website).

Database includes:

- **Metadata** → information about data (tables, columns, types, constraints, permissions, logs).
- **Stored Data** → actual values (e.g., employees' names and salaries).

Advantages of DBMS:

- **Reduces redundancy**
- **Controls access and ensures data security**
- **Ensures integrity**
- **Prevents inconsistency**
- **Provides backup & recovery**

Database Management System (DBMS) Architecture

1. **External Schema** → user/application views of data.
2. **Conceptual Schema** → complete logical design (tables, relationships, constraints).
3. **Physical Schema** → actual data storage and access methods.

- **Purpose**: **Achieve data independence** → changes at one level don't affect others
- **Data Models:**
  - **Conceptual Model** → logical structure and relationships.
  - **Physical Model** → storage and access structure.

Entity Relationship Diagram (ERD)

- **Purpose:** Used to design a database at the conceptual level.
- **Entity:** Independent object in the system.
  - Described by **attributes**.
  - Example (Banking system): *Customer* entity with attributes like `Account Number`, `Name`, `Address`.
- **Attributes:** Properties of an entity (e.g., `Student` → `ID`, `Name`, `Age`).
- **Relationships:** Define connections between entities.
  - Example: *Student* ***enrolls*** *in Course.*

- **Design Steps:**
  - Identify entities.
  - Define attributes for each entity.
  - Determine relationships between entities.
- **Goal:** Organize data effectively and ensure meaningful connections.

Relationships in Database Design

- **Definition:** Relationships represent connections between two or more entities in a database.
- **Types of Relationships**
  - **Binary - between two entities** → *Employee manages Department*
  - **Unary (Recursive) - entity relates to itself** → *Employee supervises Employee*
  - **Ternary - involving three entities** → *Employee, Project, Skill*
- **Key Concepts**
  - **Degree of Relationship:** Binary, Unary, Ternary
  - **Importance:** Proper understanding ensures correct design

---

Cardinality Ratios in Database Design

Relationship Type	Description	**Example**
1 : 1 (One-to-One)	One entity in A is related to at most one entity in B, and vice versa	Employee ↔ Contract /Employee ↔ Car
1 : N (One-to-Many)	One entity in A is related to many entities in B, but each entity in B relates to only one in A	Department → Employees /Employee → Dependents Supervisor → Employees
M : N (Many-to-Many)	Entities in A can relate to many in B, and entities in B can relate to many in A	Employee ↔ Project

Relationship Type	Description	Example
Recursive (Unary)	An entity is related to itself	Employee supervises Employee
Ternary (M : N : N)	Relationship among three entities, each side can have many	Employee ↔ Project ↔ Skill

Participation in Relationships

- **Definition:** Minimum number of relationships an entity instance must take part in.
- **Notation:**
  - **Double line** → mandatory participation
  - **Single line** → optional participation
- **Attributes on Relationships:** e.g., *Start Date* of an employee managing a department.
- **Importance in ER Design**
  - Correctly representing:
    - **Degree of relationships** (binary, ternary, etc.)
    - **Cardinality ratios** (1:1, 1:N, M:N)
    - **Participation** (mandatory vs. optional)
  - Ensures the ER diagram reflects real business rules.

ERD → Logical Design (Relational Model)

- **Tables (Relations):** Represent entities & relationships. Rows = records, Columns = attributes.

- **Primary Key:** Unique & not NULL.
- Mapping Rules
  1. Entities → Tables (Choose a suitable primary key)
  2. Composite attributes → Break into separate columns (ex. `FullName` → `FirstName`, `LastName)`
  3. Multivalued attributes → New table + FK (ex. `FullName` → `FirstName`, `LastName`).
  4. Derived attributes → Not stored, calculated when needed (ex. : `Age` → `CurrentDate`, `BirthDate`).
  5. Weak entities → Owner PK + Partial key → Composite PK (ex. `Dependent(EmployeeID + DependentName)`).
- **Goal: Ensure the logical schema is consistent, normalized, and ready for implementation in a relational DBMS.**

Comparison of Relationship Mappings

Relationship	Mapping Rule	Example
1:N	PK of “1” side → FK in “N” side	Department → Employee
Unary	FK references same table	Employee supervises Employee
1:1	PK of “may” side → FK in “must” side	Employee ↔ Department
M:N	Create bridge table (composite PK)	Employee ↔ Project (`Works_On`)
Weak Entity	Owner PK + Partial key = Composite PK	Employee → Dependent
Ternary	New table with all 3 PKs as FKs	Employee – Project – Skill