DSA5104 Principles of Data Management and Retrieval

AY2025/26 Sem1 By Zhao Peiduo

Lecture 1

Database Systems
- DBMS: interrelated data + programs; convenient and efficient environment.
- Manage data that are highly valuable, large, and concurrently accessed.
- Modern DBMSs manage large, complex collections of data; pervasive in daily life.

Database Applications — Examples
- Enterprise: Sales, Accounting, HR. Manufacturing: production, inventory, orders, supply chain.
- Banking/Finance: customers, accounts, loans/transactions; credit cards; market data.
- Universities: registration, grades. Airlines: reservations, schedules.
- Telecom: call/text/data records; bills; prepaid balances.
- Web services: online retail (tracking, recommendations), ads. Document databases.
- Navigation systems: places + routes (roads, trains, buses, ...).

Purpose of Database Systems
- Redundancy & inconsistency (multiple file formats); difficulty accessing data (new program per task).
- Data isolation (multiple files/formats) → security challenges.
- Integrity constraints buried in code → hard to add/change.
- *Atomicity*: no partial updates; e.g., fund transfer must be all-or-nothing.
- *Concurrency*: needed for performance; uncontrolled access → inconsistencies (e.g., two withdrawals).
- *Security*: restrict access to subsets of data.
- DB systems address these issues; store & retrieve data safely.

View of Data
- DB system = interrelated data + programs to access/modify.
- Provide abstract view via *data models* (concepts, relationships, semantics, constraints).

Categories of Data Models (high level)
- Relational (tables)
- Entity–Relationship (design).
- Object-based (OO/OR features).
- Semi-structured (XML/JSON).

Instances and Schemas
- *Schema*: overall design. *Instance*: data at a moment.
- Analogy: schema - variable declaration; instance - current value (class/struct blueprint vs object).

Logical vs Physical Schema & Physical Data Independence
- Logical schema: what data/relationships. Physical schema: storage layout.
- Physical data independence: change physical without changing logical; well-defined interfaces.

Data Definition Language (DDL)
- Define schema; DDL compiler → templates in data dictionary (metadata: schema, constraints, auth).
- Example: `create table instructor (ID char(5), name varchar(20), dept_name varchar(20), salary numeric(8,2))`

Data Manipulation Language (DML)
- Access/update data; *procedural* (what + how) vs *declarative* (what).
- Declarative DMLs easier; query-language part handles retrieval.

SQL Query Language
- Nonprocedural; input tables → one output table.
- Example: `select name from instructor where dept_name = 'Comp. Sci.'`
- Typically embedded or called via APIs (ODBC/JDBC); app code handles I/O/network/UI.

Engine / Components (very high level)
- Storage manager (file/buffer/authorization/transaction). Query processor (DDL interpreter, DML compiler/optimizer, eval engine).

Query Processing (stages)
- Parsing & translation Optimization Evaluation

Transaction Management
- Transaction = logical unit of work (e.g., transfer $50: read/update/write A,B).
- Ensure consistency under failures; concurrency control coordinates overlapping txns.

Architectures
- Centralized/shared-memory; Client–server; Parallel (shared-memory/disk/nothing); Distributed (geo, heterogeneity).
- App tiers: two-tier (client-DB) vs three-tier (client-app server-DB); 3-tier aids dev, scale, reliability, security.