Databases: Introduction Notes

Based on slides from Danilo Montesi

May 13, 2025

1 Study Method (Slide 2)

- **Self-study:** Focus on fundamental concepts, relate to personal experiences.
- **Exercises:** Practice is important.
- **Project/Lectures:** Develop a project or attend practical sessions using tools like DB2, SQLServer, Oracle, PostgreSQL, MySQL, MS Access, etc.

2 What is a Database? (Slides 3, 17)

- **Generic Definition:** An organized set of data that supports specific activities (within an institution, enterprise, office, or for a human).
- **Specific Definition (in the context of this course/DBMS):** A set of data *handled by a DBMS*.

3 Points of View (Slide 4)

Databases can be viewed from two perspectives:

- **Methodological:** How we design and organize data.
- **Technological:** The software and systems used to manage data.

4 Course Contents Overview (Slide 5)

- Models for organizing data.
- Languages for using data.
- Systems for handling data.
- Database design methodologies.

5 Information Systems (IS) (Slides 6, 8)

- **Definition:** A component of an institution that handles important information to pursue corporate goals.
- Each institution usually has its own IS.
- IS supports other ”subsystems” and is studied within its operational context.
- The *idea* of an Information System is independent of computer automation (existed for centuries in manual forms, e.g., demographic records, banking ledgers).

6 Handling Information Activities (Slide 7)

Information handling involves:

- Gathering, acquisition
- Storing, preservation
- Elaboration, transformation, production
- Distribution, communication, exchange

7 IT Systems (Slides 9, 10)

- **Definition:** One of the *automated parts* of the Information System. Handles information using computer technologies.
- **Hierarchy:** Enterprise $>$ Organization $>$ Information System $>$ IT System

8 Data vs Information (Slides 11-16)

- Information can be represented differently (ideas, language, drawings, numbers) and stored on various media (brain, paper, electronic).
- **In IT Systems, information is *roughly expressed as data*.**
- **Information:** Facts provided or learned about something or someone (Oxford Dictionary).
- **Data:** Pieces of information; a fixed starting point of a operation (Oxford Dictionary).
- **Key Point:** A piece of data (like a number 8) is often **useless without its "interpretation"** (context, like road signs defining hours). Data needs interpretation to become information.
- Data is the *result* of organizing, coding, and storing information (e.g., structuring a person's details into fields like Name, Surname, Tax Code).
- **Why Data is Strategic:** It's difficult to represent *rich* information/knowledge precisely. Data is a more **stable resource** compared to volatile business processes, technologies, or human roles (e.g., data in banks, demographic registries).

9 Databases Characteristics (Handled by DBMS) (Slide 18)

DBMS handles data collections that are:

- **Big:** Sizes far greater than main memory (Terabytes, billions of records).
- **Persistent:** Lifetime is independent of the computer processes using the data.
- **Shared:** Accessible by different parts/activities of an organization and multiple users.

10 Problems Without DBMS (Slide 23)

Handling data without a centralized system often leads to:

- **Redundancy:** The same data appears multiple times in different places.
- **Inconsistency:** Redundancy can cause different descriptions of the same data not to match.

11 Benefits and Properties Ensured by DBMS (Slides 18, 24-26, 30-32)

DBMS ensures:

- **Data as a Shared Resource:** Integrated resource among software applications.
  - Requires **permission mechanisms** (access control, who can read/write what).
  - Requires **concurrency checks** (managing multiple users accessing data simultaneously).
- **Privacy:** Implemented through permission mechanisms (who can access which data).
- **Reliability:**
  - Tolerant to hardware and software failures.
  - Database is a precious resource to be preserved long-term.
  - Crucial technique: **Transactions Handling**.
- **Efficiency:** Using memory and time resources profitably (response time, throughput).
- **Effectiveness:** Improves user activities by providing powerful and flexible functionalities.

12 Transactions (Slides 26, 27, 28, 29, 30, 70, 71)

- **System's Point of View:** An **unbreakable sequence of operations** (ensuring Reliability).
- **User's Point of View:** Any useful software functionality or operation, typically frequent and predefined (e.g., paying money, booking a flight). Implemented in a specific language.
- **Key Properties (leading towards ACID):**
  - **Atomic:** A set of non-detachable operations. Either *all* operations complete successfully, or *none* do ("all or nothing"). (Example: Bank transfer - withdrawal *and* deposit must both happen or neither).
  - **Concurrent:** Concurrent transactions must maintain **coherence** (e.g., preventing double booking a seat or missing a transaction on an account).
  - **Permanent Effects:** The `COMMIT` of a transaction ensures the final result is logged/tracked and persists, even with concurrency or failures.

13 DBMS vs. File System (Slide 33)

- File Systems can handle big data but provide **coarse-grained access** ("all or nothing" per file).
- **DBMS extends File System features**, providing more services and more integrated/fine-grained access to data *within* files/tables.

14 Evolution of Data Management (Slides 34-41)

- **The 70s (No DBMS):** Software applications directly accessed data stored in Files via the Operating System. Data was shared but managed independently by each app.
- **The 80s (First DBMS):** Introduction of a DBMS layer between applications and data (Data Tables, Files). DBMS provided basic data management.
- **The 90s (Procedural Behaviour / Active DBMS):**
  - **Stored Procedures:** Introduced to share common procedural logic *within* the database, callable by applications. Faced issues like standardization and impedance mismatch.

  - **Triggers:** Specific rules handled *by the DBMS itself*, often automating actions based on data events (part of "Active DBMS").
- **The 2000s (Web Architecture):** Multi-tier architecture: Client (Front-end) $\leftrightarrow$ Web Application Server (Back-end, contains application logic) $\leftrightarrow$ Database Server (DBMS, Tables, Triggers).
- **The 2010s (Mobile Apps):** Adaptation of the multi-tier architecture to include mobile clients accessing the web application server.

15 Describing Data in DBMS (Slides 42, 43)

- Software without DBMS: Internal, application-specific data descriptions $\rightarrow$ **Inconsistency**.
- DBMS: Provides a **centralized description** (catalogue or dictionary) shared by all applications $\rightarrow$ **Consistency**.
- DBMS describes data at **different levels of abstraction**.
- Data representation at higher levels is **independent from physical representation**.
- This leads to **Data Independence**: Changes in lower levels (e.g., physical storage) do not require changes in the logical schema or the software accessing the data.
- This is achieved through the **Data Model Concept**.

16 Data Model (Slide 44)

- A set of constructs used to **organize and describe the behaviour of data**.
- Crucial for providing **structure to data** (via type constructors).
- Example: The **Relational Model** provides the `relation` construct (like a table) to define sets of homogeneous records (rows).

17 Schema vs. Instance (Slides 45-47)

Using a table example (SCHEDULING):

- **Schema:** The description of the data structure (like column headers: Lectures, Lecturer, Room, Time). It is **time invariant** and describes the **intentional aspect** of the data.
- **Instance:** The actual data values in the structure (the rows of the table). It **changes rapidly** and represents the **extensional aspect** of the data.

18 Data Modelling Aspects (Slides 48-50)

- **Conceptual Modelling:** Represents data independently from any specific system. Describes **real-world concepts**. Used in the preparatory phase of a project. The most common language is **Entity-Relationship (ER) modelling**.
- **Logical Models:** Used by the DBMS to **store and organize data**. Used by software at a higher level, independent of the physical representation. Examples include **relational**, hierarchical, object, XML models.

19 Database Architecture (Simplified & ANSI/SPARC) (Slides 51, 52, 54, 55)

- **Simplified Architecture:** User → Logical Schema → Internal Schema → DB
  - **Logical Schema:** Describes database structure using the logical data model (e.g., table definitions).
  - **Internal (Physical) Schema:** Represents the logical schema at the storage level using raw data structures (files, records, pointers).
- **Standard Three-tiered ANSI/SPARC Architecture:** Users → External Schemas → Logical Schema → Internal Schema → DB
  - Adds an **External Schema** layer.
  - **External Schema:** Describes *part* of the database in a logical model (e.g., specific "views" or derived tables), potentially tailored for different users or applications.

20 Views (Slide 56)

- Views are a way to present data from one or more underlying tables in a specific format, often tailored for a particular user or application's needs (related to External Schema). Example shown combining lecture and room data.

21 Data Independence (Slides 53, 57-59)

- A key consequence of the multi-layered architecture. Data access is provided primarily through the external or logical level.
- **Physical Data Independence:** The logical and external views are independent of the physical storage representation. Changes in physical implementation (how data is stored) do not require changes in the logical schema or the software accessing the data.
- **Logical Data Independence:** The external level (views) is independent of the logical level. Changes in the logical schema (e.g., adding a column) do not necessarily require changes in the external schemas or applications using those views, provided the relevant data in the view remains available.

22 Database Languages & Interfaces (Slides 60-65)

Databases offer various ways to interact with data:

- **"Interactive" Textual Languages:** Like SQL, for direct querying and manipulation.
- **SQL Injected in Host Languages:** Embedding SQL statements within general-purpose programming languages (like Pascal, Java, C) for more complex application logic. (Slide 63 example shows embedded SQL).
- **Ad Hoc SQL / Procedural Extensions:** Languages like Oracle PL/SQL, allowing more procedural logic mixed with SQL within the database itself (e.g., stored procedures, triggers). (Slide 64 example shows PL/SQL).
- **User Friendly Interfaces (GUIs):** Graphical interfaces that abstract away textual commands for easier interaction. (Slide 65 shows an example screenshot).

23 DDL vs. DML (Slides 66, 67)

A fundamental separation in database languages:

- **DDL (Data Definition Language):** Used for **defining the schemas** (logical, external, physical) and other database structures (e.g., `CREATE TABLE`, `ALTER TABLE`, `DROP TABLE`). Slide 67 shows a `CREATE TABLE` example.
- **DML (Data Manipulation Language):** Used for **querying and updating the instances** (the actual data) within the schemas (e.g., `SELECT`, `INSERT`, `UPDATE`, `DELETE`). Slides 61/62 show `SELECT`.

24 Actors and Users (Slides 68, 69)

Different roles involved with databases:

- DBMS designers and developers
- Database designers and developers (designing specific databases)
- **DBA (Database Administrator):** A specific person or group in charge of the centralized control and handling of the database (efficiency, reliability, permissions). Often involved in design as well.
- Software designers and developers (building applications that use the database)
- **Users:**
  - **Final Users (Terminal Operators):** Execute predefined sets of operations (transactions).
  - **End/Casual Users:** Execute operations not defined a-priori using interactive languages (like SQL GUIs).

25 DBMS Pros (Slide 72)

- Data as a shared resource.
- Better modeling of the environment.
- Standardizable and scaling up centralized data management.
- Provides integrated services (security, concurrency, recovery, etc.).
- Reduces data redundancies and inconsistencies.
- Data independence supports easier software development and management.

26 DBMS Cons (Slide 73)

- Can be costly (software licenses, hardware, expertise).
- Functionalities may not be easily detachable or customizable for specific needs, potentially reducing efficiency in certain niche cases.