CS4.301: Data and Applications Monsoon 2025 8th October 2025

Tutorial 2

Topic: Data Models

What is a Data Model?

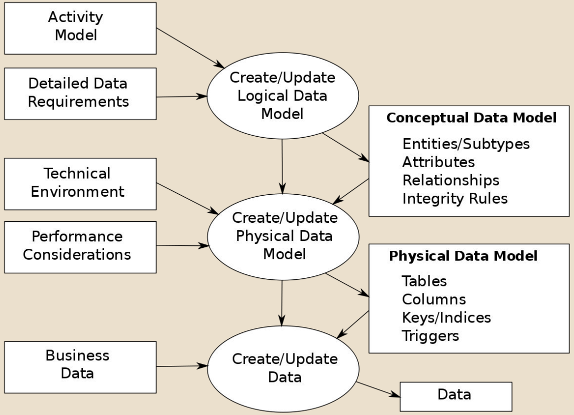
- Collection of concepts that can be used to describe the structure of a database.
- Provides the necessary means to achieve this abstraction.
- By structure of a database we mean the data types, relationships, and constraints that apply to the data.
- Think of it as the blueprint defining how data is organized, accessed, and restricted.

- Structure = data types, relationships, and constraints
- Includes basic operations (insert, delete, modify, retrieve)
- May include user-defined operations (e.g., COMPUTE_GPA on STUDENT)

Categories of data models

- High level (Conceptual)
  - They use easy to understand concepts like entities(real world objects), attributes (properties of the entities) and relationships
  - Eg: Entity-relationship model

- Representational (implementational)
  - Also called record based models
  - Commercial models
  - Easy to understand but resemble closely the actual way data is organised in storage.
  - Eg: Relational data model, network and hierarchical models



Activity Model

Detailed Data Requirements

Create/Update Logical Data Model

Conceptual Data Model

Entities/Subtypes
Attributes
Relationships
Integrity Rules

Technical Environment

Performance Considerations

Create/Update Physical Data Model

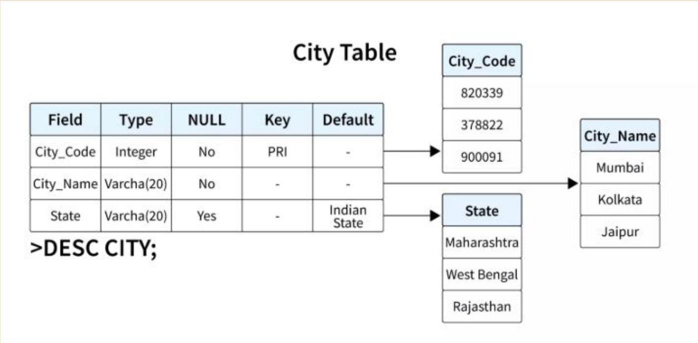
Physical Data Model

Tables
Columns
Keys/Indices
Triggers

Business Data

Create/Update Data

Data



City Table

Field	Type	NULL	Key	Default
City_Code	Integer	No	PRI	-
City_Name	Varcha(20)	No	-	-
State	Varcha(20)	Yes	-	Indian State

>DESC CITY;

City_Code
820339
378822
900091

City_Name
Mumbai
Kolkata
Jaipur

State
Maharashtra
West Bengal
Rajasthan

Categories of data models

- Low level
    - Physical models
    - Meant for computer specialists and not for end-users
    - Record formatting, record hashing, access paths

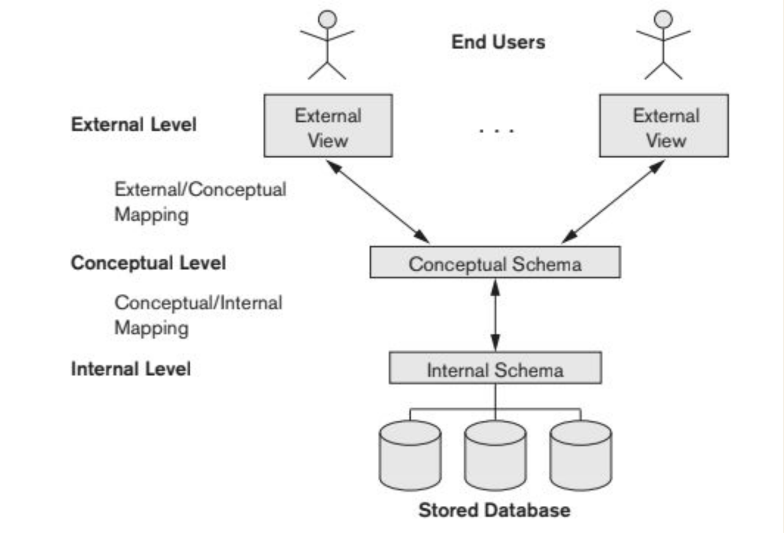
- Self-Describing
    - Combine the data description (schema) with the data values themselves
    - Eg: NoSQL, XML

Database Schemas and Instances

- Schema: Database description (structure, types, constraints)
- Specified during design, changes infrequently
- Instance/State: Actual data at a particular moment
- Changes frequently with every update
- Schema = **intension**, State = **extension**
- The DBMS is responsible for ensuring that every database state is a valid state, meaning it satisfies the schema's structure and constraints
- The DBMS stores the descriptions of the schema constructs and constraints (called the **meta-data**) in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to.

Three-Schema Architecture

- Goal: Separate user applications from physical database
- Three levels: Internal, Conceptual, External
- Achieves data independence
- Also called ANSI/SPARC architecture
- Schemas are descriptions only; data stored at physical level



Three-Schema Architecture

- **Internal Schema:**
  - Describes physical storage structures
  - Uses physical data model
  - Complete details of data storage and access paths
  - Includes indexes, file organization
  - Typically ad-hoc, system-specific
- **Conceptual Schema:**
  - Describes structure for whole database community
  - Uses representational or conceptual model
  - Hides physical storage details
  - Describes entities, data types, relationships, operations, constraints
- **External Schema:**
  - Multiple user views of database
  - Each describes part of database for specific user group
  - Hides rest of database from that group
  - Typically uses same model as conceptual schema
  - Also called view level

Data Dependence

Ability to change the schema at one level without having to change the schema at the next higher level

- Logical Data Dependence
  - Capacity to change conceptual schema without altering external schema or application programs
  - Harder to achieve

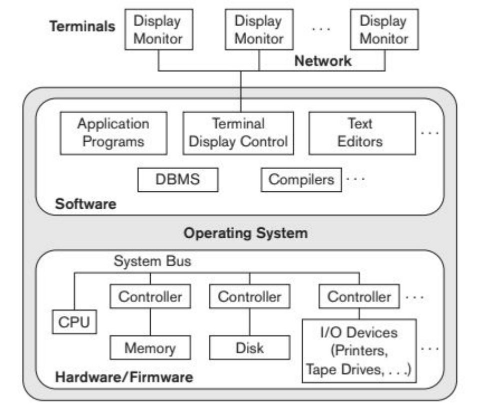
- Physical Data Dependence
  - Capacity to change internal schema without altering conceptual schema
  - Eg: reorganizing file structure

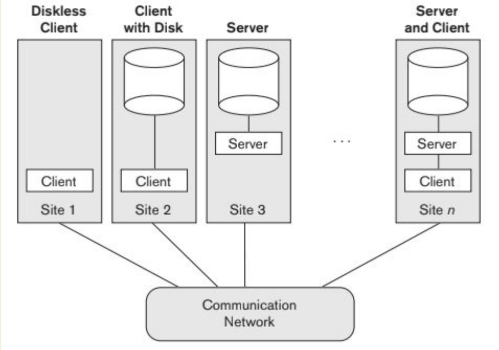
DBMS Languages

- DDL(Data Definition Language) : Used by DBAs and designers to define conceptual and internal schemas.
  - In systems with a clear separation, the DDL is used only for the conceptual schema.
- SDL(Storage Definition Language) : Used to specify the internal schema, though many modern relational DBMSs don't have a specific SDL
- VDL(View Definition Language) : Used to specify user views and their mappings to the conceptual schema. In most DBMSs, the DDL performs this role.
- DML(Data Manipulation Language): Provides a set of operations for manipulating data (retrieval, insertion, deletion, modification).
  - High Level (Non procedural)
  - Low Level (Procedural)

Centralized v/s Client/Server Architecture

- Centralized:
  - A single machine
  - Accessed using terminals
- Basic Client/Server:
  - Splits the system into client and server
  - Two-tier Client/Server:
    - The user interface and application programs are on the client side.
    - The database access functionality (query and transaction processing) is on the server side, often called a query server or SQL server
  - Three-tier Client/Server:
    - Adds an intermediate layer, the application server or Web server, between the client and the database server.





Now, your Project!

Phase 1: Requirements Gathering & Analysis

- **Goal**: Define the scope of your "mini-world" and what your database system will do.
- **Key Tasks**:
  - Describe your mini-world, its purpose, and its users.
  - Detail **Data Requirements**: The information your database must store. This includes defining entity types, weak entities, attributes, and complex relationships.
  - Detail **Functional Requirements**: The operations the system must perform, such as queries, reports, and data modifications (insert, update, delete).
- **Deliverable**: A PDF report that acts as the blueprint for the entire project.

Phase 2: Conceptual Design

- **Goal**: Create a high-level visual representation of your database structure.
- **Key Task**: Design a detailed **Entity-Relationship (ER) diagram** based on your Phase 1 requirements document.
- **Technical Focus**:
  - Use a digital tool like `draw.io` or `Lucidchart`.
  - Clearly label all entities, attributes (including primary keys, multi-valued, etc.), and relationships.
  - Specify cardinality and participation using **(min, max) notation**.
- **Deliverable**: A single-page ER diagram in a PDF file

Now, your Project!

Phase 3: Logical Design & Normalization

- **Goal**: Translate the conceptual ER diagram into a logical relational model ready for implementation.
- **Key Tasks**:
  - **Map ER to Relational Model**: Convert your entities and relationships into a set of tables (schemas) with primary and foreign keys.
  - **Normalize Your Schemas**: Refine the tables by applying normalization rules to reduce data redundancy and improve integrity. You must show the progression through **1NF, 2NF, and 3NF**.
- **Deliverable**: A PDF report with snapshots of your relational model at each stage, along with concise explanations for each conversion step

Phase 4: Implementation & Application 🚀

- **Goal**: Build the functional database and a command-line interface to interact with it.
- **Key Tasks**:
  - **Database Creation**: Use **MySQL** to write and execute SQL statements that create your tables as defined in Phase 3.
  - **Data Population**: Load the database with legitimate sample data.
  - **Application Development**: Build a **Python3** command-line interface that connects to your database.
  - **Functionality**: Implement the queries (at least 5) and updates (at least 3) from your Phase 1 requirements using raw SQL within your Python code.
- **Deliverables**: A single `.zip` file containing your Python scripts, a README with command instructions, your Phase 3 PDF, and a short video demonstration

Any doubts?