Database System Concepts and Architecture

Data model
- set of concepts to describe structure of database and certain constraints that obey.

Data Models

- **Data Model**: A set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.
- **Data Model Operations**: Operations for specifying database retrievals and updates by referring to the concepts of the data model. Operations on the data model may include *basic operations* and *user-defined operations*.

3 types of Data Models

① Conceptual
- Provides concepts that are close to the way how users perceive data.
- Also called High level or Entity-Based or Object-Based.

Categories of data models

- **Conceptual** (**high-level**, **semantic**) data models: Provide concepts that are close to the way many users *perceive* data. (Also called **entity-based** or **object-based** data models.)
- **Physical** (**low-level**, **internal**) data models: Provide concepts that describe details of how data is stored in the computer.
- **Implementation** (**representational**) data models: Provide concepts that fall between the above two, balancing user views with some computer storage details.

② Physical (Low-level, internal)
- Provides concepts of how data is stored physically in computer.

③ Implementation
- Provide concepts that fall b/w above two

Database schema → It is the description of database. The columns of database.

Database Instance
⇩
Also called
Database State
⇩
The actual data stored in db at a particular moment of time

Schemas versus Instances

- **Database Schema**: The *description* of a database. Includes descriptions of the database structure and the constraints that should hold on the database.
- **Schema Diagram**: A diagrammatic display of (some aspects of) a database schema.
- **Schema Construct**: A component of the schema or an object within the schema, e.g., STUDENT, COURSE.
- **Database Instance**: The actual data stored in a database at a *particular moment in time*. Also called **database state** (or **occurrence**).

Schema diagram
⇩
A diagram display of database Schema.

Schema construct
⇩
A component of Schema or an Object within the Schema.

Schema diagram for the database discussed earlier

STUDENT

Name	StudentNumber	Class	Major

→ Database Schema.

COURSE

CourseName	CourseNumber	CreditHours	Department

Database Schema Construct

PREREQUISITE

CourseNumber	PrerequisiteNumber

SECTION

SectionIdentifier	CourseNumber	Semester	Year	Instructor

GRADE_REPORT

StudentNumber	SectionIdentifier	Grade

Database Schema Vs. Database State

- **Database State:** Refers to the content of a database at a moment in time.
- **Initial Database State:** Refers to the database when it is loaded
- **Valid State:** A state that satisfies the structure and constraints of the database.
- **Distinction**
  - The **database schema** changes *very infrequently*. The **database state** changes *every time the database is updated.*
  - **Schema** is also called **intension**, whereas **state** is called **extension**.

Database State ⇓ Moment of db at a particular point of time.

- Initial DB state] Initial state when DB is loaded.

Valid State ⇓ A state that satisfies the structures & constraints.

When we create db we define the schema of it.

Database schema changes very rarely.

Database State changes frequently. Cause whenever we add new record state changes. but Schema doesn't.

Schema → Intension
State → Extension.

Example of a Database State

COURSE

1) Course Database at a particular moment of time.

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - **Program-data independence**.
  - Support of **multiple views** of the data.

3 Schema Architecture is introduced to support

1) Program Data independence.

2) Support multiple views of data.

3 schema ⇒ There are 3 schemas in this architecture

1) Conceptual
2) External
3) Internal.

Internal schema → Describes how & what data structures are used. Used physical data model.

Conceptual Schema
- Describes the structures & datatypes of database.
- Uses implementation data model.

Three-Schema Architecture

- Defines DBMS schemas at *three levels*:
  - **Internal schema** at the internal level to describe physical storage structures and access paths. Typically uses a *physical* data model.
  - **Conceptual schema** at the conceptual level to describe the structure and constraints for the *whole* database for a community of users. Uses a *conceptual* or an *implementation* data model.
  - **External schemas** at the external level to describe the various user views. Usually uses the same data model as the conceptual level.

External schema
- End users view of database.
- Uses some data model or conceptual level.

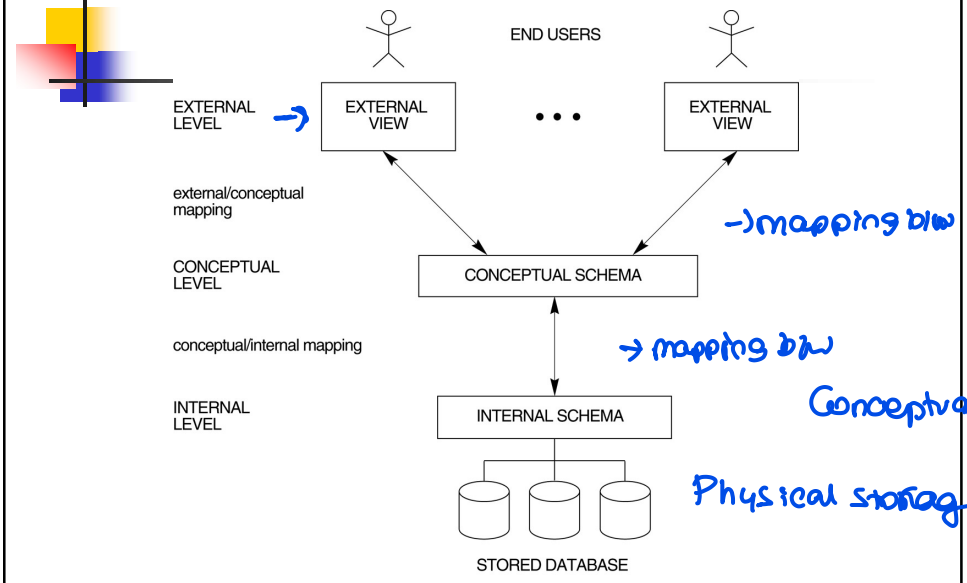
Between each of these schema levels mapping exist to process queries.

Three-Schema Architecture

Mappings among schema levels are needed to transform requests and data. Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.

Programs
↓
External Schema
↓
Mappings.

The three-schema architecture.



External Schema & Conceptual Schema

→ mapping btw

→ mapping btw

Conceptual schema & Internal schema

Physical storage

Levels of Schema

External → conceptual → internal

Physical Data Independence

- Capacity to change Internal Schema without having to change Conceptual Schema.

Logical Data Independence

- Capacity to change conceptual schema without having to check External schema and programs.

Data Independence

- **Logical Data Independence**: The capacity to change the conceptual schema without having to change the external schemas and their application programs.
- **Physical Data Independence**: The capacity to change the internal schema without having to change the conceptual schema.

- In data independence when one schema at lower level is changed the mappings b/w this schema & the higher level schemas need to be changed to supp. Data independence fully.

Data Independence

When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence. The higher-level schemas themselves are *unchanged*. Hence, the application programs need not be changed since they refer to the external schemas.

- The higher levels remain unchanged.

∴ Programs remain unchanged ∵ they refer to external schemas.

DBMS Languages

- **Data Definition Language** (**DDL**): Used by the DBA and database designers to specify the *conceptual schema* of a database. In many DBMSs, the DDL is also used to define internal and external schemas (views).

→ used to define database.
↓
i.e. structures, datatypes & constraints of DB.

In some DBMS DDL is also used to define internal & external schemas.

DML commands can be embedded in a general purpose prgm language like C

or direct DML commands can be applied through Query language like SQL.

Manipulate Database. based on queries.

DBMS Languages

- **Data Manipulation Language** (**DML**): Used to specify database retrievals and updates.
  - DML commands (**data sublanguage**) can be *embedded* in a general-purpose programming language (**host language**), such as C or an Assembly Language.
  - Alternatively, *stand-alone* DML commands can be applied directly (**query language**).

The data in DBMS can be manipulated through DML via an Interface.

User Friendly Interfaces

① Menu-based (user chooses from menu).

② Form Based (User fills data to be manipulated in a form)

DBMS Interfaces

- Stand-alone query language interfaces.
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces:
  - Menu-based, popular for browsing on the web
  - Forms-based, designed for naïve users
  - Graphics-based (Point and Click, Drag and Drop etc.)
  - Natural language: requests in written English
  - Combinations of the above

③ Natural Language: requests in written English

-Earlier everything used to happen in seperate terminals but Centralised DBMS → combines everything into one single system.
i.e- DBMS Software / Hardware / User Interface Processing Software

Centralized and Client-Server Architectures

- **Centralized DBMS:** combines everything into single system including- DBMS software, hardware, application programs and user interface processing software.

application progms etc

user interface abilities & processing.

Client-Server Architecture

-It assumes the idea where many pc's / workstations or webservers are connected to a specialised server with specific functionalities.

What is Client? It is a computer that provides

When client needs to access something about db it is connected to a server.

Server ⇒ A computer that is used to provide services to Client such as file access etc.

Basic Client-Server Architectures

- **Specialized Servers with Specialized functions**
- **Clients**
- **DBMS Server**

} ∴ Core parts of the Client Server architecture

Specialized Servers with Specialized functions:

- File Servers
- Web Servers
- E-mail Servers

} These are specialised servers with specific fxn's because many pc's / workstations etc are connected as clients to these specialised servers with specific functionalities

Clients:

- Provide appropriate interfaces and a client-version of the system to access and utilize the server resources.
- Clients maybe diskless machines or PCs or Workstations with disks with only the client software installed.
- Connected to the servers via some form of a network.
  (LAN: local area network, wireless network, etc.)

- Clients may be diskless machines or PCs or workstations etc.

- These clients are connected to servers via some form of network.

DBMS Server

- Provides database query and transaction services to the clients
- Sometimes called query and transaction servers

• Provides services to clients such as fileaccess etc.

In 2 tier client server architecture the 2 components are (i) user interface programs and application programs that run on client side.

Two Tier Client-Server Architecture

- **User Interface Programs and Application Programs** run on the client side
- Interface called **ODBC (Open Database Connectivity)** provides an Application program interface (API) allow client side programs to call the DBMS. Most DBMS vendors provide ODBC drivers.

(ii) Now if client wants to access DBMS → help of server.

Open database connectivity (ODBC)

an interface that provides an API on all client side programs to call DBMS.

-:- In b/w Client & Server there is another layer called Application Server or Web Server.

-This stores Web connectivity Software and rules & constraints part of applications.

Three Tier Client-Server Architecture

- Common for **Web applications**
- Intermediate Layer called **Application Server** or **Web Server:**
  - stores the web connectivity software and **the rules and business logic (constraints)** part of the application used to access the right amount of data from the database server
  - acts like a conduit for sending partially processed data between the database server and the client.
- **Additional Features- Security:**
  - encrypt the data at the server before transmission
  - decrypt data at the client

-It acts like a mediator b/w client & server.

Additional Features

Decrypt data at Client.

Encrypt data at Server before transmission.

Classification of DBMSs

- **Based on the data model used**:
  - Relational, Network, Hierarchical.
  - Object-oriented, Object-relational.
- **Other classifications:**
  - Single-user (typically used with micro- computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

Based on data model we use

① Relational

② Hierarchical

③ Object-oriented

④ Object-relational

Other classifications:

① Single user vs Multi user

② Centralised vs Distributed

Acknowledgement

Reference for this lecture is

- Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems,* Pearson Education.

The authors and the publishers are gratefully acknowledged.