Database System Concepts and Architecture

Data Model Schema and Instances, Database System Concept and Architecture, Data Independence, Database Language and Interfaces (DDL, DML, DCL).

Data Models

- One fundamental characteristic of the database approach is that it provides some level of data abstraction.
- **Data abstraction** generally refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data.
- **Data Model**
  - A set of concepts to describe the structure of a database, the operations for manipulating these structures, and certain constraints that the database should obey.

Data Models

❑ Data Model Structure and Constraints:

- Constructs are used to define the database structure
- Constructs typically include ***elements*** (and their ***data types***) as well as groups of elements (e.g. ***entity, record, table***), and ***relationships*** among such groups
- Constraints specify some restrictions on valid data; these constraints must be enforced at all times

❑ Data Model Operations

- These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.

- Operations on the data model may include ***basic model operations*** (e.g. generic insert, delete, update) and ***user-defined operations*** (e.g. compute_student_gpa, update_inventory)

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
- provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage.
- used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Conceptual data model

- Conceptual data models use entities, attributes, and relationships.
- An **entity** represents a real-world object or concept, such as an employee or a project.
- An **attribute** represents some property of interest that further describes an entity, such as the employee's name or salary.
- A **relationship** among two or more entities represents an association among the entities, for example, a works-on relationship between an employee and a project.
- **Entity-Relationship model**—a popular high-level conceptual data model.

Representational data model

- Representational or implementation data models are the models used most frequently in traditional commercial DBMSs.
- These include the widely used relational data model, as well as so-called legacy data models—the **network** and **hierarchical models**—that have been widely used in the past.
- Representational data models represent data by using record structures and hence are sometimes called **record-based data models**.

Schema

Database Schema

- The **description** of a database.
- Specified during database design and is not expected to change frequently.
- Includes descriptions of the database structure, data types, and the constraints on the database.

Schema Diagram

- An **illustrative** display of (most aspects of) a database schema.

Schema Construct

- A component of the schema or an object within the schema, e.g., STUDENT, COURSE.

Example of a Database Schema

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Database State

- The actual data stored in a database at a **particular moment in time**. This includes the collection of all the data in the database.
- Also called database instance (or occurrence or snapshot).
- The term *instance* is also applied to individual database components, e.g. *record instance*, *table instance*, *entity instance*.
- Refers to the **content** of a database at a moment in time.
- **Initial Database State:**
  - Refers to the database state when it is initially loaded into the system.
- **Valid State:**
  - A state that satisfies the structure and constraints of the database.

Example of a database state

COURSE

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

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Database Schema vs. Database State

- The database schema changes very infrequently.
- The database state changes every time the database is updated.
- Schema is also called **intension**.
- State is also called **extension**.

Questions

Q1: List the characteristics of DBMS

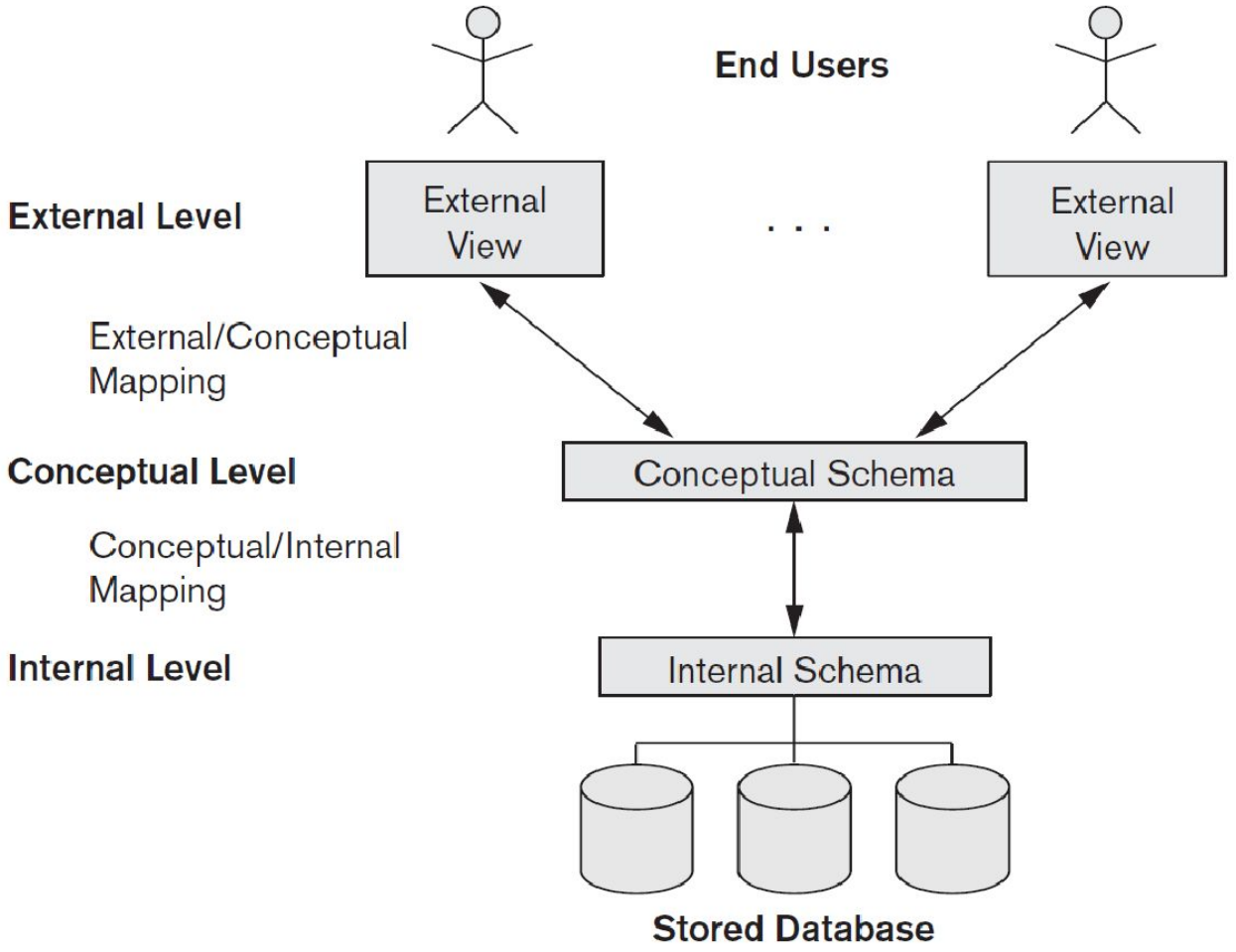
Q2: What is Data Models?

Q3: What is a Database State?

Q4: What is schema?

Three-Schema Architecture

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



Three-Schema Architecture

- The goal of the three-schema architecture is to separate user applications from the physical database.
- Defines DBMS schemas at **three** levels:
  - **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
    - 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.
    - hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
    - Uses an **implementation** data model.
  - **External schemas** at the external level to describe the various user views
    - Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.
    - Usually uses the same data model (representational data model) as the conceptual schema.

Three-Schema Architecture

- Three schemas are only descriptions of data; the stored data that actually exists is at the physical level only. Each user group refers to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view.
- The processes of transforming requests and results between levels are called **mappings**.
- 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.
  - Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

Data Independence

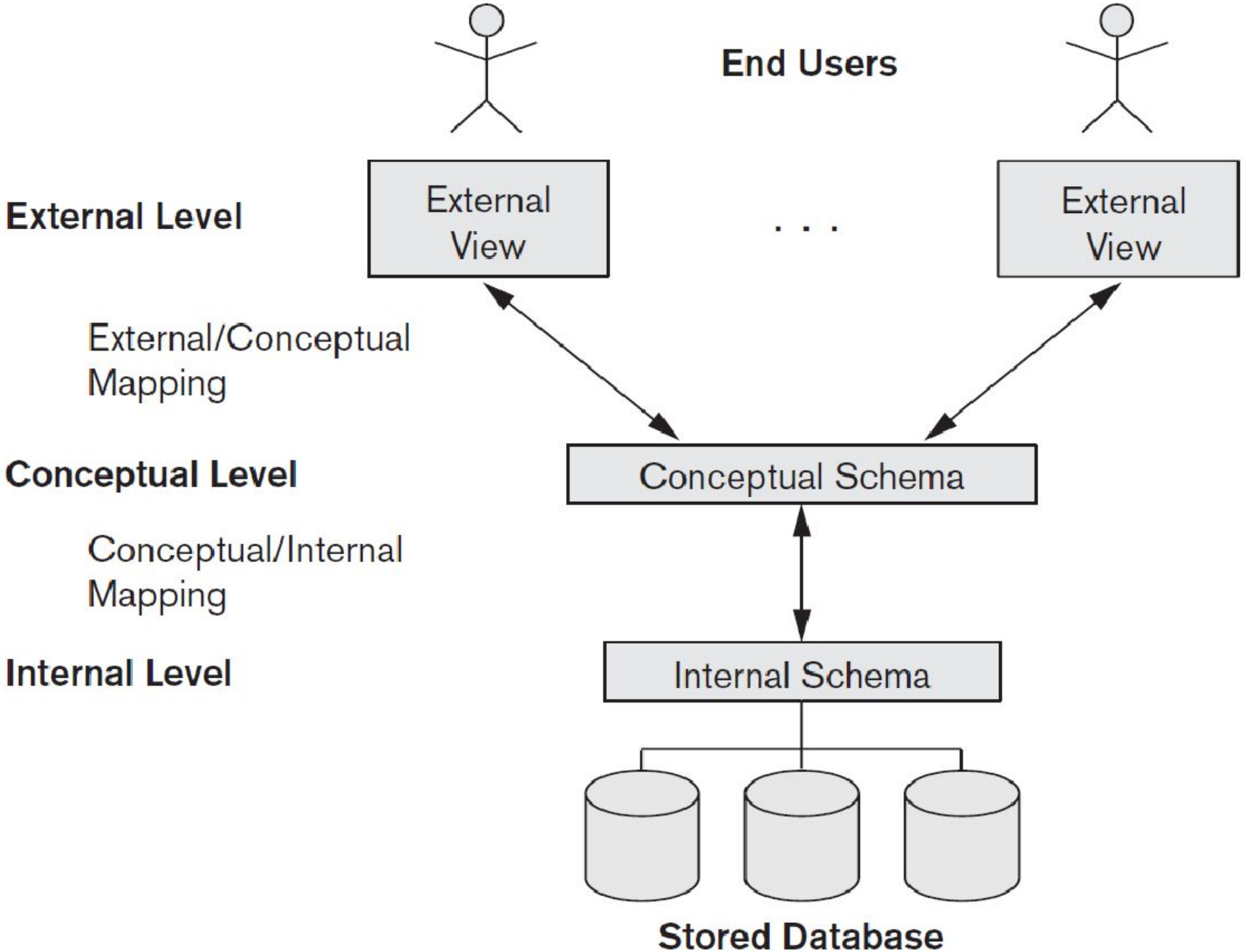
- Capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

Logical Data Independence

- The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.

Physical Data Independence

- The capacity to change the internal schema without having to change the conceptual schema.
- For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance


End Users
External Level
External View
. . .
External View
External/Conceptual Mapping
Conceptual Level
Conceptual Schema
Conceptual/Internal Mapping
Internal Level
Internal Schema
Stored Database

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.

DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
  - High-Level or Non-procedural Languages: These include the relational language SQL
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural Languages:
    - These must be embedded in a programming language

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 external schemas (views).

DBMS Languages

❑ Data Manipulation Language (DML)

- Used to specify database retrievals and updates
- DML commands can be *embedded* in a general-purpose programming language (host language), such as C, C++, or Java.
  - A library of functions can also be provided to access the DBMS from a programming language
- Alternatively, stand-alone DML commands can be applied directly (called a query language).

Types of DML

❑ High Level or Non-procedural Language

- Are “set”-oriented (can specify and retrieve many records in a single DML statement) and specify *what* data to retrieve rather than *how* to retrieve it.
- Also called **declarative** languages.
- For example, a query in SQL

❑ Low Level or Procedural Language

- Retrieve data one *record-at-a-time.*
- Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

❑ A high-level DML used in a standalone interactive manner is called a **query language**.

DBMS Interfaces

- Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL Plus in ORACLE)
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
  - Menu-based, forms-based, graphics-based, etc.
- Mobile Interfaces: interfaces allowing users to perform transactions using mobile apps

DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
  - **Embedded Approach:** e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)
  - **Procedure Call Approach:** e.g. JDBC for Java, ODBC for other programming languages. **ODBC** is an SQL-based Application Programming Interface (API) created by Microsoft that is used by Windows software applications to access databases via SQL. **JDBC** is an SQL-based API created by Sun Microsystems to enable Java applications to use SQL for database access.
  - **Database Programming Language Approach:** e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components.
  - **Scripting Languages:** Server-side scripting languages such as PHP and Python are used to write database programs.

DBMS Interfaces

❑ User-Friendly DBMS Interfaces

- Menu-Based Interfaces for Web Clients or Browsing
- App for mobile devices
- Forms-based, designed for users used to filling in entries on a form.
- Graphical User Interfaces: A GUI typically displays a schema to the user in diagrammatic form.
  - Point and Click, Drag and Drop, etc.
  - Specifying a query on a schema diagram
- Natural language: requests in written English
- Keyword-based Database Search
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces

DBMS Interfaces

❑ Interfaces for the DBA

- Creating user accounts, granting authorizations
- Setting system parameters
- Changing schemas or access paths

❑ Interfaces for Parametric Users

- Bank tellers, small set of operation that they perform repeatedly
-