DATABASE MANAGEMENT SYSTEM (DBMS)

CONTENT

- Three-Schema Architecture-Mapping
- Data Independence
- Logical Data Independence
- Physical Data Independence
- Difference between Logical and Physical Data Independence
- Data model Schema and Instance
- Database Schema vs. Database State

Three-Schema Architecture-Mapping

- 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

- **Applications insulated from how data is structured and stored.**
- **Data independence is the capacity to change the schema at one level of the architecture without having to change the schema at the next higher level.**
- We distinguish between **logical** and **physical** data independence according to which two adjacent levels are involved.
- **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.

Logical Data Independence

- **Logical Data Independence-** Ability to change the conceptual schema without changing external schemas or application programs.
  - **Refers to immunity of external schemas to changes in conceptual schema.**
  - **Conceptual schema changes (e.g. addition/removal of entities).**
  - **Should not require changes to external schema or rewrites of application programs**
  - Example: adding a field to a table should not affect other users view of the data

Physical Data Independence

- **Physical Data Independence-** Ability to change the internal (physical) schema without changing the conceptual schema.
  - **Refers to immunity of conceptual schema to changes in the internal schema.**
  - **Internal schema changes (e.g. using different file organizations, storage structures/devices).**
  - **Should not require change to conceptual or external schemas.**
  - Example: moving physical files from one disk to another. Easier to implement than logical independence.
- An **example of physical data independence**
  - suppose that the internal schema is modified (because we decide to add a new index, or change the encoding scheme used in representing some field's value, or stipulate that some previously unordered file must be ordered by a particular field). Then we can change the mapping between the conceptual and internal schemas in order to avoid changing the conceptual schema itself.

Difference between Logical and Physical Data Independence

- **Physical Data Independence**
  - Protection from changes in physical structure of data.
  - It is the ability to modify the physical schema without causing application programs to be rewritten.
  - In other words, old programs do not have to be rewritten, when changes are made to physical storage structure or the physical devices on which data are stored.
- **Logical Data Independence:**
  - Protection from changes in logical structure of data.
  - It is the ability to modify the conceptual schema without causing application program to be rewritten.
  - Logical data independence is more difficult to achieve than physical data independence, since program are having dependence the logical structure of the database.

Data model Schema and Instance

- The overall design of a database is called schema.
- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - e.g., the database consists of information about a set of customers and accounts and the relationship between them
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level

  - A database may also have several schemas at the view level, sometimes called **subschemas**, that describe different views of the database.

Database Schemas and Types

- Database Schema:
  - The ***description*** of a database.
  - 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.

Database Schema

- A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- A database schema can be represented by using the visual diagram. That diagram shows the database objects and relationship with each other.
- A database schema is designed by the database designers to help programmers whose software will interact with the database.
- The process of database creation is called data modeling.

Database Schema

- A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema diagram.
- For example, the given figure neither show the data type of each data item nor the relationship among various files.
- In the database, actual data changes quite frequently.
- For example, in the given figure, the database changes whenever we add a new grade or add a student. The data at a particular moment of time is called the instance of the database.

Instances

- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an **instance** of the database.
  - Example:
  - A program written in a programming language. A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an *instance* of a database schema.

Database State:

- 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*

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 state when it is initially loaded into the system.
- 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**.
- **State** is also called **extension**.

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

Figure 2.1
Schema diagram for the database in Figure 1.2.

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

Figure 1.2
A database that stores student and course information.