Lecture-3
(Introduction to Databases)

©Dr. Nikhil Tripathi
https://nikhiltripathi.in

What will we learn in this lecture?

- Feel free to test yourself ☺
- Data Abstraction
- Three level data abstraction
- Instances
- Schema
- Database schemas
- Data Independence

Self Assessment

- What is a database?
  - Collection of data containing information relevant to an enterprise.
- What is DBMS?
  - Will discuss in this lecture.
- Goal of DBMS:
  - Storing and retrieving database information conveniently and efficiently.
- Disadvantages of File-processing system:
  - Data redundancy and inconsistency.
  - Security problems.

View of Data

- Database system:
  - A collection of interrelated data
    - e.g. University -> Departments, Students, etc., IPL Team -> Owner, Players, Sponsors, etc.
  - A set of programs to access and/or modify the stored data
    - e.g. Insertion of data for a newly joined faculty member in university database,
    - Deletion of a injured player's data from an IPL team database
  - Purpose of DBMS
    - Avoid unnecessary complexities (data storage and maintenance)
      - Users should have just an abstract view of the data.

Data Abstraction

- How can we define a database system as usable?
  - It must retrieve data efficiently
  - The interaction must be user-friendly
  - The user must receive only the required data (Nothing more, nothing less): Security aspect of a database system
- All these constraints require usage of complex data structures to represent data in the database
- A simple database system user is not necessarily equipped with computer knowledge.
- As a result, DB developers hide the complexity from users through several abstraction levels: Physical Level, Logical Level and View Level
- Thus, the process of hiding irrelevant details from user is called data abstraction

Data Abstraction Levels

- 3-level data abstraction architecture
- Physical Level
  - Lowest level of abstraction
  - *How* is the data actually stored in memory?
  - Involvement of complex low-level data structures
- Logical Level
  - Middle level of data abstraction architecture
  - *What* data are stored in the database and what relationships exist among those data
    - E.g. In a university database, a faculty falls under academic section (table) while a fee personnel falls under accounts section.
    - E.g. In an IPL team database, a player falls under 'Playing team' section while a sponsor falls under 'sponsors list' section.

Data Abstraction Levels

- View Level
  - Highest level of abstraction
  - Describes only part of the entire database
  - Complexity may still exist because of the variety of information stored in a large database.
  - A user may not need all this data. Instead, s/he may need only a part of it.
    - E.g. A fee personnel in the university database only needs to see how much fee a particular student has paid. He may not require the student's details such as his marks in IDB.
  - This level exists to simplify the DB user's interaction with the system by providing variety of views.
    - E.g. Students and the amount of tuition fee; student and the status (hosteller, non-hosteller, etc.)

Data Abstraction Levels

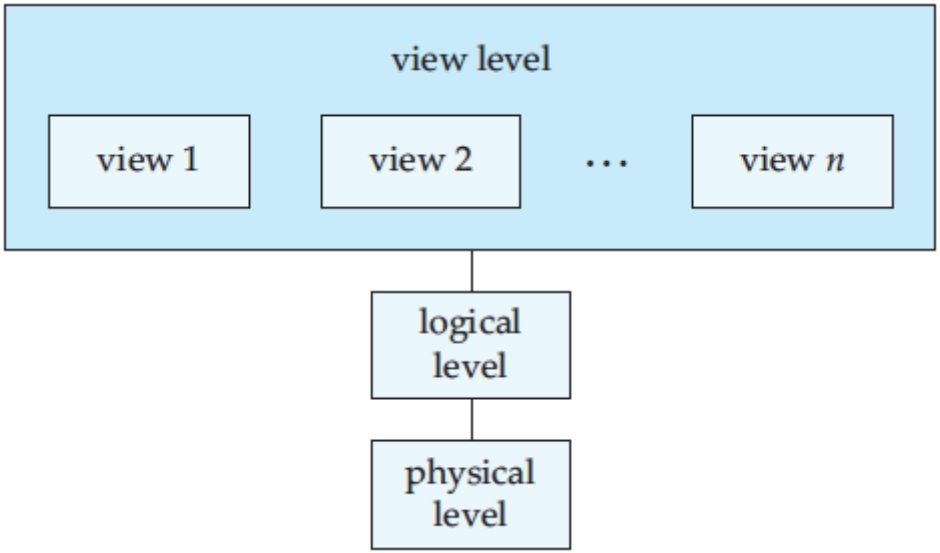

Image source: Silberschatz, Korth and Sudharsan (All copyrights are owned by the source)

Interview question: Why is there different versions of *views* at View Level?

Data Abstraction Levels

- Consider an example where a record is to be stored into a table.
- The record:
  - type *instructor* = **record**
    - *ID*: **char**;
    - *name*: **char**;
    - *salary*: **numeric**;
    - end;
- Record ‘instructor’ is having three fields (attributes).

Data Abstraction Levels

- At physical level:
  - This record is described as a block of consecutive storage locations (may be few bytes or kilobytes) in the hard disk.
  - The compiler hides this detail from the DB programmers/administrators.
- At logical level:
  - This record is stored as fields and attributes along with their data types and their relationship. DB administrators usually work at this level.
- At view level:
  - A DB user (e.g. a fee personnel in accounts section) just interacts using GUI by putting a student's unique ID and retrieving the fee details. S/he may be restricted from accessing the academic performance of the student.

Instances and Schemas

- Data stored within a database change over time as new information is added and old information is deleted.
  - Passing out final year batch and admission of first year batch in a university – Accounts database need to be updated.
- The collection of information stored in a database at a particular moment is called as an *instance* of the database.
- The overall design of the database is called the database *schema*. Schemas are changed infrequently unlike instances.

Instances and Schemas

- Example: Analogy to a program in a programming language.
- A database schema corresponds to the variable declarations in a program.
  - The variables used in the program are rarely changed in a stable program/software.
- However, the value of a variable keeps changing as the program executes.
  - Thus, a variable has different values at different instances of time.
- Summary: Schema is changed rarely while the instance is changed frequently.

Database Schemas and Data Independence

- A database system has several schemas, partitioned according to the levels of abstraction.
- Physical schema: Database design at the physical level
- Logical schema: Database design at the logical level
- View schemas: A database may have several schemas at the view level. These are referred to as subschemas.
  - Subschemas describe different views of the database.
    - E.g. student_ID, fee_amount_paid to an account section employee while student_ID, courses_registered to an academic section employee

Database Schemas and Data Independence

- Logical schemas are the most important schemas since DB programmers construct the applications by using the logical schema.
- Since physical schema is hidden beneath the logical schema, it can be changed easily without affecting application programs.
- Application programs are said to exhibit physical data independence if they do not depend on the physical schema, and thus need not be rewritten if the physical schema changes.

Lecture Summary

- Data Abstraction
- Three data abstraction levels - physical level, logical level and view level.
- Examples specific to these abstraction levels.
- Instances and Schemas.
- Database schemas – physical schema, logical schema and view schemas.
- Physical Data Independence.

References

- Silberschatz, Korth and Sudharsan