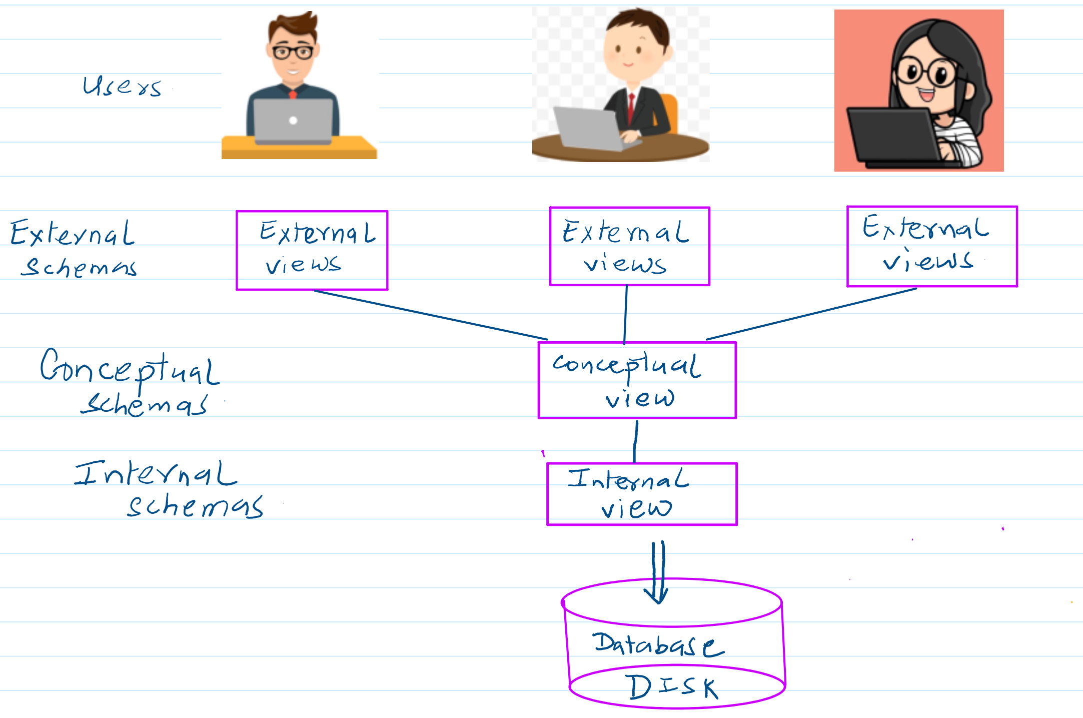
Introduction to DBMS part-2

11 August 2024 11:38

View of Data
A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data. A major purpose of database system is to provide users with an abstract view of data. That is the system hides certain details of how the data are stored and maintained.

Data abstraction
For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database systems, users are not computer trained. Developers hide the complexity from users through several levels of abstractions to. simplify users interactions with the system:



Levels of Abstraction in a DBMS.

- **Physical level or Internal view or schema:** The lowest level of abstraction describes how the data are actually stored. The physical level describes complex, low level data structures in detail.
- **Logical level or conceptual level or view or schema:** The next higher level of abstraction describes what data are stored in the database and what relationship exist among those data The logical level thus describes the entire database in terms of small numbers of relatively simple structures, although implementation of the simple structures at the logical level may involve complex physical level structures. The users of the logical level does not need to be aware of this complexity. This is referred to as **physical data independence**. Database Administrator, who must decide what information to keep in the database use the logical level of design or abstraction.
  An analogy to the concept of data types in programming languages may clarify the distinction among levels of abstraction. Many high level programming language supports the notion of a structured type. For example, we may describe a record as follows:
  Class written in C++:

```
class Student{
   int rollNumber;
   string course;
   int courseId;
   string name;
```

```
                    int semNum;
                };
```
This code defines a new record type called student with 5 fields. Each field has a name and a type associated with.

A university organization may have such record types, including

- Department with fields dept_name, building, and budget
- Course with fields course_id, Title, department_name and credits.
- Students with fields, ID, name, dept_name, and tot_cred
- Instructor with fields, ind_id, dept_name, and salary

Add to physical level an instructor a department or a student can be described as a block of consecutive storage locations. The compiler hides this level of detail from programmers. Similarly, the database system hides many of the lowest level storage details from database programmers. Database Administrator, on the other hand, may be aware of certain details of the physical organization of the data.

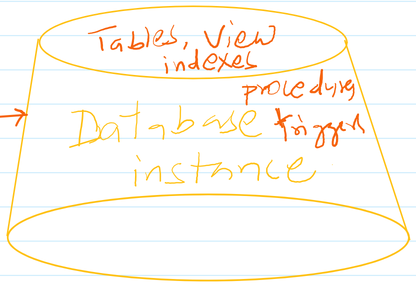
At the logical level, each such a record is described by a type definition. As in the previous code segment and the interrelationship of these record types is defined as well. Programmers using a programming language, work at this level of abstraction. Similarly, database administrators usually work at this level of Abstraction.

Finally, at the view level, computer users see a set of application programs that hide details of the data types. At the view level, several views of the database are defined, and database users sees some or all of these views. In addition to hiding details of the logical level of the database, the views also provide a security mechanism to prevent users from accessing certain parts of the database. For example, clerks in the university registrar office can see only that part of the database that has information about student. They cannot access information about salaries of instructors.

$\boxed{\text{SSD}}$

Instances and Schemas

State of this object changes over time. $\rightarrow$

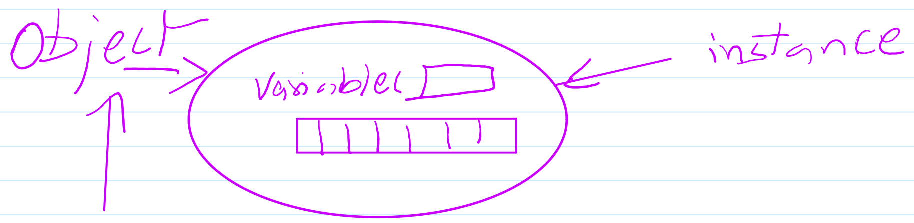


Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an instance of a database. The overall design of the database is called the database schema. Schemas are changed infrequently, if at all. The concept of database schemas and instances can be understood by an analogy of a program written in a programming language. A database schema corresponds to a variable declaration in a program.

Each variable has a particular value at a given instant. The values of the variable in a program at a point in time correspond to an instance of a database schema. Database systems have several schemas partition according to the levels of abstraction. The physical schema describes the database design at the **physical level**, while the **logical schema** describes the database design at the **logical level**. A database may also have several schemas at the view level, sometimes called **sub schemas** which described different views of the database. Of these, the **logical schemas** is by far the most important in terms of its effect on application programs. Since programmers construct applications by using the **logical schema**, the **physical schema** is hidden beneath the logical schema, and can usually be changed easily without affecting application programs. Application programs are set to exhibit physical data independence if they do not depend on the physical schema. And thus need not to be rewritten if the physical schema changes.

```
int i;
for(i=0; i<10; i++){        ←   i=0
```

```
int i;
for(i=0; i<10; i++){   ← i=0
```



Change its state during the execution of code, that is why it is called an instance.

Data Models
Underlying the structure of database is the **data model**. A collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints. A data model provides a way to describe the design of the database at the physical, logical and view levels.

- **Relational model:** The relational model you. uses a collection of tables to represent both data and the relationship among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations The relational model is an example of a record based model.

- **Entity-relationship model:** The entity relationship (E-R.) data model uses a collection of basic objects called entities and relationship among these objects. An entity is a "thing" or "object" in the real world, that is distinguishable from other objects. The entity relationship model is widely used in database design.

- **Object-oriented Data model:** Object oriented programming, especially in Java, C, Python or C Sharp(C#). has become dominant software development methodology. This led to a development of an object oriented data model that can be seen as extending the er model with notions of encapsulation methods and object identity. The object relational data model combines features of the object oriented data model and relational data model.

- **Semi-structured Data model:** The semi structured data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The extensive markup language (XML) or (JSON) JavaScript Object Notation is widely used to represent semi-structured data.
  JSON Syntax:
  ```
  {
     "Key": value,
  },
  {
     "Key": value
  }
  ```

  https://www.json.org/json-en.html

  The above is the official website of JSON.

Data models present in research level(not in real life use):
- Network model

- Hierarchical model

Database Languages
A database system provides a data definition language to specify the database schema and a data manipulation language to express database queries and updates.
In practice, the data definition and data manipulation languages are not two separate language. Instead, they simply form parts of a single database language such as the widely used SQL language.

Data- Manipulation Language(DML)
A data manipulation language or (DML) is a language that enables user to access or manipulate data as organized by the appropriate data model, the type of access are:

- Retrieval of information stored in a database.
- Insertion of a new information into the database.
- Deletion of information from the database.
- Modification of information stored in the database.

There are basically two types:

- **Procedural DML's :** We require a user to specify what data are needed and how to get those data.
- **Declarative DML's:** Require a user to specify what data are needed without specifying how to get those data.

Query: A query is a statement requesting the retrieval of information a portion of a DML, that involves information retrieval is called a query language. although technically incorrect, it is common practice to use the term query language and data manipulation language synonymously.

Data-definition language(DML)
We specify database schema by a set of definition expressed by your special language called **a data-definition language, or (DDL).** The DDL also used to specify additional properties of the data.
We specify the storage of structure and access method used by the database system by a set of statements in a special type of ddl called a data storage and definition language. These statements define the implementation details of the database schema, which are usually hidden from the users. The data values stored in the database must satisfy a certain consistency constraints. For example, suppose the university requires that the account balance of a department must never be negative. The DDL provide facilities to specify such constraints. The database system checks these constraints every time the database is updated In general, a constraint can be an arbitrary predicate pertaining to the database. How. arbitrary predicates may be costly to test thus. database system implement integrity constraints that can be tested with a minimal overhead.

Domain Constraints:
A domain of possible values must be associated with every attribute For example, integer types, character types, date or time types. Declaring an attribute to be of a particular domain acts as a constraint on the values that it can take. Domain constraints are most elementary form of integrity constraints. They are tested easily by the system whenever a new data item is entered into the database.

output range

$\sin x \ [-\infty, \infty] \qquad [-1, 1]$

input Domain

Referential Integrity:
There are cases where we wish to ensure that a value that appears in one relation for a given set of attributes also appear in a certain set of attributes in another relation (A referential integrity.) For example, the department listed for each course must be one that actually exists. More precisely, the department name value in a course record must appear in the department name attribute of some record of the department relation. Database modification can causeway violation of referential integrity. When referential integrity constraint is violated, the normal procedure is to reject the action that caused the violation.

Assertions: An assertion is any condition that the. database must always satisfy. domain constraints and referential integrity constraints are special forms of assertions. However, there are many constraints that we cannot express by using only these special forms. For example, every department must have at least five forces offered every semester must be expressed as an assertion when an assertion is created, the system tests it for validity if the assertion is valid, then any future modification to the database is allowed only if it does not Causeway the assertion to be violated.

Authorization: We may want to differentiate among the users as far as the type of access they are permitted on various data values in the database. These differentiation are expressed in terms of authorization. The most common being **read authorization**, which allows reading but not modification of data. **Insert authorization**, which allows insertion of new data, but not modification of existing data. **Update authorization**, which allows modification, but not deletion of data and **delete authorization**, which allows deletion of data. We may assign the user all, none, or a combination of these type of authentication or authorization.

Homework
Read and write paragraph on meta-data.
Write 10 points on Database Administrator and Database users
Everything in msword file from everyone.