**Database management system**

Module 1: part 2

**Database system concepts and architecture**

D

ata model is collection of concepts that can be used to escribe the structure of a database—provides the necessary means to achieve this abstraction.

Types of data model

1.High level or conceptual data model:

* provide concepts that are close to the way many users perceive data

2.Low level or physical data model:

* provide concepts that describe the details of how data is stored on the computer storage media, typically magnetic disks.
* Concepts provided by low-level data models are generally meant for computer specialists, not for end users

Schemas

* The description of a database is called the **database schema**, which is specified during database design and is not expected to change frequently.
* A displayed schema is called a **schema diagram**
* Each object of the schema is called **schema construct**
* The data in the database at a particular moment in time is called a **database state** or snapshot.
* The schema is sometimes called the **intension**, and a database state is called an **extension** of the schema.

**The three-schema architecture**:

The goal of the three-schema architecture, is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

1. The **internal level** has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
2. The **conceptual level** has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.
3. The **external or view level** includes a number of external schemas or 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. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

Note: The processes of transforming requests and results between levels are called **mappings**

**Data independence:**

The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the **capacity to change the schema at one level of a database system without having to change the schema at the next higher level**.

1. **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs
2. **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema.

**Database language interfaces**:

* data definition language (**DDL**), is used by the DBA and by database designers to define both schemas.
* The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.
* In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only.
* Another language, the storage definition language (**SDL**), is used to specify the internal schema.
* The mappings between the two schemas may be specified in either one of these languages.
* In most relational DBMSs today, there is no specific language that performs the role of SDL.
* Instead, the internal schema is specified by a combination of functions, parameters, and specifications related to storage.
* These permit the DBA staff to control indexing choices and mapping of data to storage.
* For a true three-schema architecture, we would need a third language, the view definition language (**VDL**), to specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.
* In relational DBMSs, **SQL** is used in the role of VDL to define user or application views as results of predefined queries
* Typical manipulations include retrieval, insertion, deletion, and modification of the data. The DBMS provides a set of operations or a language called the data manipulation language (**DML**) for these purposes
* **High level DML**: High level DMLs, such as SQL, can specify and retrieve many records in a single DML statement; therefore, they are called **set-at-a-time or set-oriented DMLs**. A query in a high-level DML often specifies which data to retrieve rather than how to retrieve it; therefore, such languages are also called **declarative**.
* **Low level DML**: This type of DML typically retrieves individual records or objects from the database and processes each separately. Low-level DMLs are also called **record-at-a-**time DMLs because of this property

**DBMS interfaces**:

* Menu-Based Interfaces for Web Clients or Browsing.
* Form based interface
* Graphical user interface
* Natural Language Interfaces
* Speech Input and Output.
* Interfaces for Parametric Users.
* Interface for DBA

**DBMS component model**:

1. User type: DBA staff

* The database and the DBMS catalog are usually stored on disk.
* Access to the disk is controlled primarily by the operating system (OS), which schedules disk read/write.
* Many DBMSs have their own buffer management module to schedule disk read/write, because this has a considerable effect on performance.
* Reducing disk read/write improves performance considerably. A higher-level stored data manager module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.
* The DBA staff works on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands.
* The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.
* The catalog includes information such as the names and sizes of files, names and data types of data items, storage details of each file, mapping information among schemas, and constraints.
* In addition, the catalog stores many other types of information that are needed by the DBMS modules, which can then look up the catalog information as needed.

1. User type: casual or naïve user

* Casual users and persons with occasional need for information from the database interact using some form of interface, which we call the interactive query interface
* These queries are parsed and validated for correctness of the query syntax, the names of files and data elements, and so on by a query compiler that compiles them into an internal form.
* This internal query is subjected to query optimization Among other things, the query optimizer is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution.
* It consults the system catalog for statistical and other physical information about the stored data and generates executable code that performs the necessary operations for the query and makes calls on the runtime processor

1. User type: application programmers

* Application programmers write programs in host languages such as Java, C, or C++ that are submitted to a pre-compiler.
* The pre compiler extracts DML commands from an application program written in a host programming language.
* These commands are sent to the DML compiler for compilation into object code for database access.
* The rest of the program is sent to the host language compiler.
* The object codes for the DML commands and the rest of the program are linked, forming a canned transaction whose executable code includes calls to the runtime database processor.
* Canned transactions are executed repeatedly by **parametric users**, who simply supply the parameters to the transactions.
* Each execution is considered to be a separate transaction.
* An example is a bank withdrawal transaction where the account number and the amount may be supplied as parameters.

1. runtime database processor

* (1) the privileged commands,
* (2) the executable query plans, and
* (3) the canned transactions with runtime parameters.
* It works with the system catalog and may update it with statistics.
* It also works with the stored data manager, which in turn uses basic operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory.
* The runtime database processor handles other aspects of data transfer, such as management of buffers in the main memory.
* Some DBMSs have their own buffer management module while others depend on the OS for buffer management.
* concurrency control and backup and recovery systems are integrated into the working of the runtime database processor for purposes of transaction management

**Database system utilities**

**DBMS** will have database utilities to help the DBA to manage the database system.

* **Loading**: A loading utility is used to load existing data files—such as text files or sequential files—into the database. Usually, the current format of the data file and the desired (target) database file structures are specified to the utility, which then automatically reformats the data and stores it in the database.
* **Backup**: A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium. The backup copy can be used to restore the database in case of catastrophic disk failure
* **Database storage reorganization**: This utility can be used to reorganize a set of database files into different file organizations, and create new access paths to improve performance
* **Performance monitoring**: Such a utility monitors database usage and provides statistics to the DBA.

**Centralised and Client or server architectures for DBMSs**

* **Centralised DBMS architecture**: database systems used these computers similarly to how they had used display terminals, so that the DBMS itself was still a centralized DBMS in which all the DBMS functionality, application program execution, and user interface processing were carried out on one machine
* **Basic client /server architecture:**

**T**he resources provided by specialized servers can be accessed by many client machines. The client machines provide the user with the appropriate interfaces to utilize these servers, as well as with local processing power to run local applications.

**A** server is a system containing both hardware and software that can provide services to the client machines, such as file access, printing, archiving, or database access.

* **Two-tier** **client /server architecture:**

The different approach to two-tier client/server architecture was taken by some object-oriented DBMSs, where the software modules of the DBMS were divided between client and server in a more integrated way

The architectures described here are called two-tier architectures because the software components are distributed over two systems: client and server. The advantages of this architecture are its simplicity and seamless compatibility with existing systems. The emergence of the Web changed the roles of clients and servers, leading to the three-tier architecture

* **Three-tier** **client /server architecture:**

Many Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server. This intermediate layer or **middle tier** is called the application server or the Web server.

This server plays an intermediary role by running application programs and storing business rules (procedures or constraints) that are used to access data from the database server. It can also improve database security by checking a client’s credentials before forwarding a request to the database server.

**Classification of database management system**

1. The basic **relational data model** represents a database as a collection of tables, where each table can be stored as a separate file. Most relational databases use the high-level query language called SQL and support a limited form of user views
2. The **object data model** defines a database in terms of objects, their properties, and their operations. Objects with the same structure and behaviour belong to a class, and classes are organized into hierarchies (or acyclic graphs). The operations of each class are specified in terms of predefined procedures called methods.
3. Relational DBMSs have been extending their models to incorporate object database concepts and other capabilities; these systems are referred to as **object-relational** or extended relational systems
4. The **XML model** has emerged as a standard for exchanging data over the Web, and has been used as a basis for implementing several prototype native XML systems. XML uses hierarchical tree structures. It combines database concepts with concepts from document representation models. Data is represented as elements; with the use of tags, data can be nested to create complex hierarchical structures. This model conceptually resembles the object model but uses different terminology. XML capabilities have been added to many commercial DBMS products
5. The **network model** represents data as record types and also represents a limited type of 1:N relationship, called a set type. A 1:N, or one-to-many, relationship relates one instance of a record to many record instances using some pointer linking mechanism in these models
6. The **hierarchical model** represents data as hierarchical tree structures. Each hierarchy represents a number of related records. There is no standard language for the hierarchical model.

* ………………………..end