Unit – 1 Database system introduction

**Introduction :**

A *database*  is an organized collection of facts. In other words we can say that it is a collection of information arranged and presented to serve an assigned purpose.

An example of a database is a dictionary, where words are arranged alphabetically. Another example is a telephone directory, where subscriber names are listed in an alphabetic order. Similarly, when we think of a box of cards with names and addresses written as a mailing, list, then the box and its contents form a mailing database. Contains cards with mailing addresses. All these cards are placed in alphabetic order of names and the collection of these cards would be called a database.

In order to keep database updated, we may need to perform operations like adding information, removing information, editing existing information, etc. take the case of a telephone diary, in which we note down the names, addresses and phone numbers of our friends. This also a type database. If we make a new friend, we note the information about him/her in the diary which is equivalent to adding data to a database. If address of our friend changes then we note the new address in our diary which means *editing* a database. If we want to send a letter to our friend then we look into the diary to locate the address. This is called *searching* a database

In case our telephone diary contains a very large number of names, addresses, phone numbers. Moreover, the names are not arranged in any order. Thus, it would be very difficult to locate the names, addresses, and phone number of our friends. Thus, it is essential to arrange the names in some order, say alphabetically, to make the search easy. However, as the number of friends gets large, managing the database manually becomes difficult. A database management software package is a helpful tool in such a situation.

**Data and Information**

**Data** - People sometimes use the terms data and information interchangeably, but data and information are different. Data can be text, numbers, audio, video, images, or any combination of these. There many ways to collect data, including surveys, interviews, the use of sensors, the reading of documents, and even the monitoring of brain waves, sophisticated void-activated technology is already available that allows people to store data by simply speaking into a computer.

**Information : when** we process data and covert it into a from that is useful and meaningful to the decision maker, it becomes **information.** Human beings apply facts, principles, knowledge, experience, and intuition to convert data into information. Only then does it become useful for making decisions. Note, however, that it is difficult to place a dollar value on information, also, information is time-dependent, since its value and usefulness often decrease with time.

**Database:**

A **database** is the collection of related persistent data and contains information relevant to an

enterprise. The database is also called the repository or container for a collection of data files.

For example, **university database** for maintaining information about students, courses and

grades in university.

**Database Management System:**

A **database-management System(DBMS)** is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the **database,** contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient.*

**In other words, DBMS,** **is a software package that allows data to be effectively stored, retrieved and manipulated. The data contained in a DBMS package can be accessed by multiple application and users. Applications packages such as SQL Server, Oracle, My – SQL, MS-Access are commercial available DBMS packages.**

A DBMS is usually a very large software package that carries out many different tasks including the provision of facilities to enable the user to access and modify information in the database. The database is an intermediate link between the physical database, the various facilities of different types of users, a DBMS normally provides one or more specialized programming languages often called database language. Different DBMS provide different database languages although a language called SQL has recently taken on the role of a de facto standard

**Relational database management system-** The relational model was introduced in an academic paper by E.F. Codd in 1970 as a way to make database management systems more independent of any particular way to make database management systems more independent of any particular application.

The products that are generally referred to as relational databases in fact implement a model that is only an approximation to the mathematical model defined by Codd. Three key terms are used extensively in relational database models: relations, attributes, and domains. A relation is a table with columns and rows. The named columns of the relation are called attributes, and the domain is the set of values the attributes are allowed to take.

**T**he relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. the relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

|  |  |  |
| --- | --- | --- |
| **Supplier number** | **Supplier name** | **Supplier address** |
| S001 | Pradeep | Kathmandu-5 |
| S002 | Binay | Kathmandu-6 |
| S003 | Khemo | Nawalparasi-2 |
| S004 | Dhiraj | Bharatpur-10 |

|  |  |  |  |
| --- | --- | --- | --- |
| **PART NUMBER** | **PART DESCRIPTION** | **UNIT PRINCE** | **SUPPLIER NUMBER** |
| P001 | Door Latch | 22.20 | S001 |
| P002 | Door handle | 25.15 | S002 |
| P003 | Compressor | 27.45 | S003 |
| P004 | Door stopper | 10.35 | S004 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ord\_Number** | **Order\_Date** | **Delivery Date** | **Part Number** | **Part Amount** | **Order Total** |
| O001 | 11.11.2007 | 21.11.2007 | P001 | **2** | **44.40** |
| O002 | 12.11.2007 | 22.11.2007 | P002 | **3** | **75.45** |
| O003 | 13.11.2007 | 23.11.2007 | P003 | **4** | **109.80** |
| O004 | 14.11.2007 | 24.11.2007 | P004 | **5** | **53.75** |

In the above figure, a Supplier Table, a Part Table, and an Order Table. In each table, the rows are unique records and the columns are fields. Another term for a row or record in a relation is a **tuple**. Often a user needs information from a number of relations to produce a report. Here is the strength of the relational model. It can relate data in any one file or table to data in another file or table as long as both tables share a common data element.

**Advantage :** The relational model has many advantages, but its most appealing quality is its grate simplicity.

**First,** users can easily related to tables and hence find the data structure in a relational model can tables and hence find the data structure in a relational model easy to understand and implement.

**Second,** users are not burdened with issues such as storage structure and access strategy because a relational database automatically addresses these issues.

**Third,** the relational model is flexible and can integrate data and information from multiple files.

**Fourth,** the relational model supports ad hoc queries, this feature is important if users are to be able to use the full power of the database.

**Fifth,** in a relational model, new data elements, can easily be added and old ones deleted or updated without any significant design changes to the database.

**Disadvantage :** there are some drawbacks in the Relational database model.

**First:** it is slow by comparison with other data models, since it has to access data from different tables and this can become tedious.

**Second,** if database is large, the data redundancy problem occurs, same data can be stored in several tables.

**Advantages of Database Systems (DBMS's)**   
  
The Database Systems provide the following advantages over the traditional file system.

**1) Controlled redundancy**: In a traditional file system, each application program has its own data, which causes duplication of common data items in more than one file. This duplication/redundancy requires multiple updations for a single transaction and wastes a lot of storage space. We cannot eliminate all redundancy due to technical reasons. But in a database, this duplication can be carefully controlled, that means the database system is aware of the redundancy and it assumes the responsibility for propagating updates.   
**2) Restricting Unauthorized Access**

When multiple users share a database, it is likely that some users will not be authorized to access all information in the database. For example, financial data is often considered confidential, and hence only authorized persons are allowed to access such data. In addition, some users may be permitted only to retrieve data, whereas others are allowed both to retrieve and to update. Hence, the type of access operation—retrieval or update—must also be controlled. Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database. A DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions.

**3) Providing Persistent Storage for Program Objects and Data Structures**

Databases can be used to provide persistent storage for program objects and data structures. This is one of the main reasons for the emergence of the object-oriented database systems. Programming languages typically have complex data structures, such as record types in PASCAL or class definitions in C++. The values of program variables are discarded once a program terminates, unless the programmer explicitly stores them in permanent files, which often involves converting these complex structures into a format suitable for file storage. When the need arises to read this data once more, the programmer must convert from the file format to the program variable structure. Object-oriented database systems are compatible with programming languages such as C++ and JAVA, and the DBMS software automatically performs any necessary conversions. Hence, a complex object in C++ can be stored permanently in an object-oriented DBMS,

**4) Providing Multiple User Interfaces**

Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces. These include query languages for casual users; programming language interfaces for application programmers; forms and command codes for parametric users; and menu-driven interfaces and natural language interfaces for stand-alone users. Both forms-style interfaces and menu-driven interfaces are commonly known as graphical user interfaces (GUIs).

5) **Data consistency**: The problem of updating multiple files in traditional file system leads to inaccurate data as different files may contain different information of the same data item at a given point of time. This causes incorrect or contradictory information to its users. In database systems, this problem of inconsistent data is automatically solved by controlling the redundancy.   
  
6) **Program data independence**: The traditional file systems are generally data dependent, which implies that the data organization and access strategies are dictated by the needs of the specific application and the application programs are developed accordingly. However, the database systems provide an independence between the file system and application program, that allows for changes at one level of the data without affecting others. This property of database systems allow to change data without changing the application programs that process the data.   
  
7) **Sharing of data**: In database systems, the data is centrally controlled and can be shared by all authorized users. The sharing of data means not only the existing applications programs can also share the data in the database but new application programs can be developed to operate on the existing data. Furthermore, the requirements of the new application programs may be satisfied without creating any new file.   
  
8) **Enforcement of standards**: In database systems, data being stored at one central place, standards can easily be enforced by the DBA. This ensures standardized data formats to facilitate data transfers between systems. Applicable standards might include any or all of the following departmental, installation, organizational, industry, corporate, national or international.   
  
9) **Improved data integrity**: Data integrity means that the data contained in the database is both accurate and consistent. The centralized control property allow adequate checks can be incorporated to provide data integrity. One integrity check that should be incorporated in the database is to ensure that if there is a reference to certain object, that object must exist.   
  
10) **Improved security**: Database security means protecting the data contained in the database from un - authorized users. The DBA ensures that proper access procedures are followed, including proper authentically schemes for access to the DBMS and additional checks before permitting access to sensitive data. The level of security could be different for various types of Data and operations.   
  
11) **Data access is efficient**: The database system utilizes different sophisticated techniques to access the stored data very efficiently.   
  
12) **Conflicting requirements can be balanced**: The DBA resolves the conflicting requirements of various users and applications by knowing the overall requirements of the organization. The DBA can structure the system to provide an overall service that is best for the organization.   
  
15) **Improved backup and recovery facility**: Through its backup and recovery subsystem, the database system provides the facilities for recovering from hardware or software failures. The recovery subsystem of the database system ensures that the database is restored to the state it was in before the program started executing, in case of system crash. 

**Disadvantages of DBMS**

In spite of the advantages of using a DBMS, there are a few situations in which such a system mayinvolve unnecessary overhead costs as that would not be incurred in traditional file processing. The overhead costs of using a DBMS are due to the following:

• High initial investment in hardware, software, and training.

• Generality that a DBMS provides for defining and processing data.

• Overhead for providing security, concurrency control, recovery, and integrity functions.

Additional problems may arise if the database designers and DBA do not properly design the database or if the database systems applications are not implemented properly. Hence, it may be more desirable to use regular files under the following circumstances:

• The database and applications are simple, well defined, and not expected to change.

• There are stringent real-time requirements for some programs that may not be met because of DBMS overhead.

• Multiple-user access to data is not required.

**Database System Application:**

Databases are widely used. Here are some representative applications:

**● Banking :-** for customer information, accounts, and loans and banking transactions.

**● Airlines :** for reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner-situated around the world accessed the central database system through phone lines and other data networks.

**● Universities :-** for student information, course registrations, and grades.

**● Credit card transactions :-** for purchases on credit cards and generations of monthly statements.

**● Telecommunications:-** for keeping records of calls made, generating monthly bills maintaining balances on prepaid calling cards and storing information about the communication networks.

**● Finance :-** for storing information about holdings, sales and purchases of financial instruments such a stocks and bonds.

**● Manufacturing :-** for management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores and order for items.

**Data Models:** Underlying the structure of a 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 a database at the physical, logical, and view level.

The data model can be classified in four different categories:

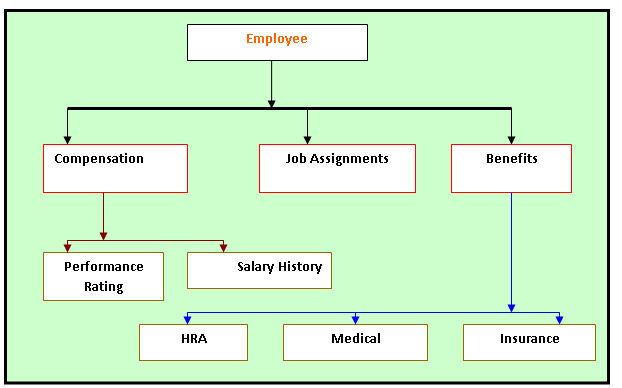
● **Relational Model :-** The relational model was introduced in an academic paper by E.F. Codd in 1970 as a way to make database management systems more independent of any particular way to make database management systems more independent of any particular application.

The products that are generally referred to as relational databases in fact implement a model that is only an approximation to the mathematical model defined by Codd. Three key terms are used extensively in relational database models: relations, attributes, and domains. A relation is a table with columns and rows. The named columns of the relation are called attributes, and the domain is the set of values the attributes are allowed to take.

|  |  |  |
| --- | --- | --- |
| **Supplier number** | **Supplier name** | **Supplier address** |
| S001 | Pradeep | Kathmandu-5 |
| S002 | Binay | Kathmandu-6 |
| S003 | Khemo | Nawalparasi-2 |
| S004 | Dhiraj | Bharatpur-10 |

|  |  |  |  |
| --- | --- | --- | --- |
| **PART NUMBER** | **PART DESCRIPTION** | **UNIT PRINCE** | **SUPPLIER NUMBER** |
| P001 | Door Latch | 22.20 | S001 |
| P002 | Door handle | 25.15 | S002 |
| P003 | Compressor | 27.45 | S003 |
| P004 | Door stopper | 10.35 | S004 |

● **Hierarchical Data Model :** In a hierarchical model, the logical relationships among various data elements are represented as a hierarchy. In the hierarchical data model presents data to users in a treelike structure. The most common hierarchical DBMS is IBM’s .within each record, data elements are organized into pieces of records called segments. To the, each record looks like an organization chart with one top-level segment called root. An upper segment is connected logically to a lower segment in a parent-child can have only one parent.



● **Network Data Model :** The network data model is a variation of the hierarchical data model. Indeed, database can be translated from hierarchical to network and vice versa in order to optimize processing speed and convenience. Whereas, hierarchical structures depict one to many relationships, network structures depict data logically as many-to-many relationships. In other words, parents can have multiple “children” and a child can have more than one parent.



● **Entity-relationship (E-R) model** is a *high level* data model based on a perception of a real world that consists of collection of basic objects, called **entities**, and of **relationships** among these entities. An **entity** is a thing or object in the real world that is distinguishable from other

objects. Entities are described in a database by a set of **attributes**. A **relationship** is an

association among several entities. The set of all entities of the same type is called an **entity set**

and the set of all relationships of the same type is called a **relationship** *set*. Overall logical

structure of a database can be expressed graphically by E-R diagram. The basic components of

this diagram are:

o **Rectangles** (represent entity sets)

o **Ellipses** (represent attributes)

o **Diamonds** (represent relationship sets among entity sets)

o **Lines** (link attributes to entity sets and entity sets to relationship sets)



The figure shown above is an example of E-R diagram.

● **Object-Based Data Model :** The object-oriented data model is another data model that has seen increasing attention. The object-orientation model can be seen as extending the E-R model with notions of encapsulation, method and object identity. The object - relational data model combines features of the object-oriented data model and relational data model

**Database Languages:**

The main objective of a database management system is to allow its users to perform a number of operations on the database such as insert, delete, and retrieve data in abstract terms without knowing about the physical representations of data. To provide the various facilities to different types of users, a DBMS normally provides one or more specialized programming languages called [Database](http://my.safaribooksonline.com/9788131731925/gloss01" \l "gloss01_066) (or DBMS) Languages.

The DBMS mainly provides two database languages, namely, [data definition language](http://my.safaribooksonline.com/9788131731925/gloss01" \l "gloss01_058) and [data manipulation language](http://my.safaribooksonline.com/9788131731925/gloss01" \l "gloss01_062) to implement the databases. Data definition language (DDL) is used for defining the database schema. The DBMS comprises DDL compiler that identifies and stores the schema description in the DBMS catalog. Data manipulation language (DML) is used to manipulate the database.

#### ● Data Definition Language(DDL)

In DBMSs where no strict separation between the levels of the database is maintained, the data definition language is used to define the conceptual and internal schemas for the database. On the other hand, in DBMSs, where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only. In such DBMSs, a separate language, namely, ***storage definition language (SDL)*** is used to define the internal schema. Some of the DBMSs that are based on true three-schema architecture use a third language, namely, ***view definition language (VDL)*** to define the external schema.

The DDL statements are also used to specify the integrity rules (constraints) in order to maintain the integrity of the database. The various integrity constraints are domain constraints, referential integrity, assertions and authorization. These constraints are discussed in detail in subsequent chapters. Like any other programming language, DDL also accepts input in the form of instructions (statements) and generates the description of schema as output. The output is placed in the **data dictionary**, which is a special type of table containing metadata. The DBMS refers the data dictionary before reading or modifying the data. Note that the database users cannot update the data dictionary; instead it is only modified by database system itself.

CREATE    - It is used to create schema of table in the database.  
  
ALTER     - It is used to change the schema of table in  the database.  
  
DROP      - It is used to delete the table or databse objects from the database.  
  
TRUNCATE  - It removes all records from a table, including all spaces allocated for the records are removed.  
  
COMMENT   - add comments to the data dictionary.  
  
RENAME    - rename an database object or schema of a table.

#### ● Data Manipulation Language

Once the database schemas are defined and the initial data is loaded into the database, several operations such as retrieval, insertion, deletion, and modification can be applied to the database. The DBMS provides data manipulation language (DML) that enables users to retrieve and manipulate the data. The statement which is used to retrieve the information is called a **query**. The part of the DML used to retrieve the information is called a **query language**. However, query language and DML are used synonymously though technically incorrect. The DML are of two types, namely, non-procedural DML and procedural DML.

The **non-procedural** or **high-level** or **declarative DML** enables to specify the complex database operations concisely. It requires a user to specify what data is required without specifying how to retrieve the required data. For example, SQL (**Structured Query Language**) is a non-procedural query language as it enables user to easily define the structure or modify the data in the database without specifying the details ofhow to manipulate the database. The high-level DML statements can either be entered interactively or embedded in a general purpose programming language. On the other hand, the **procedural** or **low-level DML** requires user to specify what data is required and how to access that data by providing step-by-step procedure. For example, relational algebra is procedural query language, which consists of set of operations such as select, project, union, etc., to manipulate the data in the database.

SELECT - It will show desired data in the form of a table from the database.  
  
INSERT - It is used to insert data values in a table of the database.  
  
UPDATE - It is used for update existing data values within a table of the database.  
  
DELETE - It is used to delete all records from a table, the space for the records remain in the database.  
  
MERGE  - UPSERT operation (insert or update)  
  
CALL - call a PL/SQL or Java subprogram  
  
EXPLAIN PLAN - explain access path to data  
  
LOCK TABLE - control concurrency

● **TCL - Transaction Control (TCL)** statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.  
  
COMMIT - This command will save work done in the table   
  
SAVEPOINT - This command will identify a point in a transaction to which you can later rollback..  
  
ROLLBACK - This command will restore database to original point since the last COMMIT  statement executed  
  
SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use.

**Database Schema and Stance**

In any data model, it is important to distinguish between the *description* of the database

and the *database itself*. The description of a database is called the **database schema**, which is specified during database design and is not expected to change frequently. Most data models have certain conventions for displaying schemas as diagrams. A displayed schema is called a **schema diagram**. A schema diagram for the database shown in Figure 1.2; the diagram displays the structure of each record type but not the actual instances of records. We call each

object in the schema—such as STUDENT or COURSE—a **schema construct**.

schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints. Other aspects are not specified in the schema diagram; for example, Figure 2.1 shows neither the data type of each data item, nor the relationships among the various files. Many types of constraints are not represented in schema diagrams. A constraint such as students majoring in computer science must take CS1310 before the end of their sophomore year is quite difficult to represent diagrammatically.

The actual data in a database may change quite frequently. For example, the database shown in Figure 1.2 changes every time we add a new student or enter a new grade. The data in the database at a particular moment in time is called a database state or snapshot. It is also called the current set of occurrences or instances in the database. In a given database state, each schema construct has its own current set of instances; for example, the STUDENT construct will contain the set of individual student entities (records) as its instances. Many database states can be constructed to correspond to a particular database schema. Every time we insert or delete a

record or change the value of a data item in a record, we change one state of the

database into another state. The DBMS stores the descriptions of the schema constructs and constraints—also called the meta-data—in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to. The schema is sometimes called the intension, and a database state is called an extension of the schema.

|  |
| --- |
|  |

**Data Abstraction:**

Data abstraction is the technique of hiding the complexity of the database to its users. There are three levels of data abstraction which are discussed below.

• **Physical Level or Internal Level:**

It is the lowest level of abstraction and describes *how* the data in the database are actually stored. This level describes complex low-level data structures in detail and is concerned with the way the data is physically stored. Data only exists at physical level.

• **Logical Level or Conceptual Level:**

This is the next higher level of abstraction and describes *what* data are stored in the database, and what relationships exist among those data. It describes the structure of whole database and hides details of physical storage structure. It concentrates on describing entities, data types, relationships, attributes and constraints. All of the views must be derivable from this conceptual schema.

• **View Level or External Level:**

It is the highest level of abstraction and is concerned with the way the data is seen by individual users. This level simplifies the users’ interaction with the system. It includes a number of user views and hence is guided by the end user requirement. It describes only those part of the database in which the users are interested and hides rest of all from those users. Each user group refers to its own external schema.

Example:

view level

• View result

• View student information

logical level: entire database schema

• Courses (CourseNo,CourseName,Credits,Dept)

• Student (StudentID,Lname,Fname,Level,Major)

• Grade (StudentID,CourseNo,mark)

**physical level:**

• how these tables are stored, how many bytes it required etc.

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

database. The process of transforming requests and results between levels is called mapping.

|  |
| --- |
|  |

**Data Independence**

A database is viewed through any three levels of abstraction. Any change at any level may not affect other level’s scheme. As the databases keep growing, there may be changes made at some level. However, this should never database. The concept of data independence program beneficial in such a context.

*The ability to modify a schema definition in one level without affecting a schema definition in the next higher level is called* ***data independence.***

There are two levels of data independence. These are follows:

1. Physical
2. Logical
3. **physical data independence :** physical Data independence refers to the ability to modify the schema followed at the physical level without affecting the schema followed at a conceptual level. That is, the application programs remain the same even though the schema at a physical level gets modified.

*Modification at the physical level are occasionally necessary in order to improve performance of the system.*

1. **Logical Data Independence :-** Logical Data independence refers to the ability to modify the conceptual schema without causing any changes in the schema followed at view level. The logical data independence ensures that the application programs remain the same. Modifications at the conceptual level are necessary whenever logical structures of the database get altered because of some unavoidable reasons. Such a case may be the introduction of maternity leave for female employees in the employee table.

***The abstract data types in modern programming languages implement the concept of data independence to a larger extent. Both types hide implementation details from the users. this allows users to concentrate on the general structure rather than low-level implementation details.***

**DATABASE USERS:**

The people who use the database can be categorized

a) **Database users**

b) **Database administrator (DBA).**

**a) Database users are of 4 different types:**

**1) Native users:**

These are the unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. E.g. consider a user who checks for account balance information over the World Wide Web. Such a user access a form, enters the account number and password etc. And the application program on the internet then retrieves the account balance using given account information which was passed to the user.

**2*)* Application programmers:**

These are computer professionals who write application programs, used to develop user interfaces. The application programmer uses Rapid Application Development (RAD) toolkit or special type of programming languages which include special features to facilitate creation of forms and display of date on screen.

**3) Sophisticated users:**

These users interact with the database using database query language. They submit their query to the query processor. Then Data Manipulation Language (DML) functions are performed on the database to retrieve the data. Tools used by these users are OLAP(Online Analytical Processing) and data mining tools.

**4) Specialized users:**

These users write specialized database applications to retrieve data. These applications can be used to retrieve data with complex data types e.g. graphics data and audio data.

**b) Database Administrator (DBA)**

A person having who has central control over data and programs that access the data is called DBA. Following are the functions of the DBA.

Centralized control of the database is exerted by a person or group of persons under the supervision of a high-level administrator. This person or group is referred to as the database administrator (DBA). They are the users who are most familiar with the database and are responsible for creating, modifying, and maintaining its three levels.

Database Administrator is responsible to manage the DBMS’s use and ensure that the database is functioning properly.

**There are. Following are the functions of the DBA.**

**1) Schema definition:** DBA creates database schema by executing Data Definition Language (DDL) statements.

**2) Schema and physical organization modification:** If any changes are to be made in the original schema, to fit the need of your organization, then these changes are carried out by the DBA.

**3) Granting of authorization for data access:** DBA can decide which parts of data can be accessed by which users. Before any user access the data, dbms checks which rights are grantd to the user by the DBA.

**4) Routine maintenance:** DBA has to take periodic backups of the database, ensure that enough disk space is available to store new data, ensure that performance of dbms is not degraded by any operation carried out by the users.

5) **Deciding the information content of the database**

It is the DBA’s job to decide exactly what information is to be held in the database - in other words, to identify the entities of interest to the enterprise and to identify the information to be recorded about those entities. Having done this, the DBA must then define the content of the database by writing the conceptual schema.

**6) Deciding the storage structure and access strategy**

The DBA must also decide how the data is to be represented in the database, and must specify the representation by writing the storage structure definition. In addition, the associated mapping between storage structure definition and the conceptual schema must also be specified.

**7) Liaising with the users**

It is the business of the DBA to liaise with users, to ensure that the data they require is available, and to write the necessary external schemas. In addition, the mapping between any given external schema and the conceptual schema must also be specified. In practice the external DDL will probably include the means for specifying the mapping, but the schema and the mapping should be clearly distinguishable.

**8) Defining authorization checks and validation procedures**

Authorization checks and validation procedures may be considered as logical extensions of the conceptual schema. The conceptual DDL will include facilities for specifying such checks and procedures.

**9) Defining a strategy for backup and recovery**

Once an enterprise is committed to a database, it become critically dependent on the successful operation of that system. In the event of damage to any portion of the database – caused by human error, say, or a failure in the hardware or supporting operating system – it is essential to be able to repair the data concerned with a minimum of delay and with as little effect as possible on the rest of the system.

**10) Monitoring performance and responsibilities to changes in requirements**

The DBA is responsible for so organizing the system as to get the performance that is “best for the enterprise” and for making the appropriate adjustments change. Any change to details of storage and access must be accompanied by a corresponding change to the definition of the mapping to storage, so that the conceptual schema may remain constant.

**DATABASE APPLICATION ARCHITECTURE**

A DBMS is a complex software system. Software components that constitute a DBMS and the types of computer system software with which the DBMS interacts.

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

**Two-Tier Client/Server Architectures for DBMSs**

In relational database management systems (RDBMSs), many of which started as centralized systems, the system components that were first moved to the client side were the user interface and application programs. Because SQL provided a standard language for RDBMSs, this created a logical dividing point between client and server. Hence, the query and transaction functionality related to SQL processing remained on the server side. In such an architecture, the server is often called a **query server** or **transaction server** because it provides these two functionalities. In an RDBMS, the server is also often called an **SQL server**.

The user interface programs and application programs can run on the client side. When DBMS access is required, the program establishes a connection to the DBMS (which is on the server side); once the connection is created, the client program can communicate with the DBMS. A standard called **Open Database Connectivity** (**ODBC**) provides an **application programming interface** (**API**), which allows client-side programs to call the DBMS, as long as both client and server machines have the necessary software installed. Most DBMS vendors provide ODBC drivers

for their systems. A client program can actually connect to several RDBMSs and send query and transaction requests using the ODBC API, which are then processed at the server sites. Any query results are sent back to the client program, which can process and display the results as needed. A related standard for the Java programming language, called **JDBC**, has also been defined. This allows Java client programs to access one or more DBMSs through a standard interface.

## 3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

|  |
| --- |
|  |

* **Database (Data) Tier** − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
* **Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
* **User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

**Classification of Database Management Systems**

Several criteria are normally used to classify DBMSs. The first is the **data model** on which the DBMS is based. The main data model used in many current commercial DBMSs is the **relational data model**. The **object data model** has been implemented in some commercial systems but has not had widespread use.Many legacy applications still run on database systems based on the **hierarchical** and **network data models**. Examples of hierarchical DBMSs include IMS (IBM) and some other systems like System 2K (SAS Inc.) and TDMS. IMS is still used at governmental and industrial installations, including hospitals and banks, although many of its users have converted to relational systems. The network data model was used by many vendors and the resulting products like IDMS (Cullinet—now Computer Associates), DMS 1100

(Univac—now Unisys), IMAGE (Hewlett-Packard).

The relational DBMSs are evolving continuously, and, in particular, have been incorporating many of the concepts that were developed in object databases. This has led to a new class of DBMSs called object-relational DBMSs.We can categorize DBMSs based on the data model: relational, object, object-relational, hierarchical, network, and other.

More recently, some experimental DBMSs are based on the XML (eXtended Markup Language) model, which is a tree-structured (hierarchical) data model. These have been called native XML DBMSs. Several commercial relational DBMSs have added XML interfaces and storage to their

products.

The second criterion used to classify DBMSs is the number of users supported by the system. Single-user systems support only one user at a time and are mostly used with PCs. Multiuser systems, which include the majority of DBMSs, support concurrent multiple users.

The third criterion is the number of sites over which the database is distributed. A DBMS is centralized if the data is stored at a single computer site. A centralized DBMS can support multiple users, but the DBMS and the database reside totally at a single computer site. A distributed DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites, connected by a computer network. Homogeneous DDBMSs use the same DBMS software at all the sites, whereas heterogeneous DDBMSs can use different DBMS software at each site. It is also possible to develop middleware software to access several autonomous

preexisting databases stored under heterogeneous DBMSs.