

Database Management Systems

UNIT -1

- 1.0 Introduction and brief history to Database
- 1.1 Characteristics of database
- 1.2 Difference between File System & DBMS.
- 1.3 Advantages of DBMS
- 1.4 Functions of DBMS
- 1.5 Role of Database Administrator
- 1.6 Simplified Database System Environment
- 1.7 Example of a Database
- 1.8 Architecture of DBMS
- 1.9 Data Independence
- 1.10 Types of database applications
- 1.11 Data Models
- 1.12 The database system environment
- 1.13 Centralized and Client-Server DBMS Architectures

Introduction to Database

1.0 Introduction

Database is a collection of related data. Database management system is software designed to assist the maintenance and utilization of large scale collection of data. DBMS came into existence in 1960 by Charles. Integrated data store which is also called as the first general purpose DBMS. Again in 1960 IBM brought IMS-Information management system. In 1970 Edgar Codd at IBM came with new database called RDBMS. In 1980 then came SQL Architecture- Structure Query Language. In 1980 to 1990 there were advances in DBMS e.g. DB2, ORACLE.

Data

- Data is raw fact or figures or entity.
- When activities in the organization takes place, the effect of these activities need to be recorded which is known as Data.

Information

- Processed data is called information
- The purpose of data processing is to generate the information required for carrying out the business activities.

In general data management consists of following tasks

- Data capture: Which is the task associated with gathering the data as and when they originate.
- Data classification: Captured data has to be classified based on the nature and intended usage.
- Data storage: The segregated data has to be stored properly.
- Data arranging: It is very important to arrange the data properly
- Data retrieval: Data will be required frequently for further processing,
Hence it is very important to create some indexes so that data can be retrieved

easily.

- Data maintenance: Maintenance is the task concerned with keeping the data up-to-date.
- Data Verification: Before storing the data it must be verified for any error.
- Data Coding: Data will be coded for easy reference.
- Data Editing: Editing means re-arranging the data or modifying the data for presentation.
- Data transcription: This is the activity where the data is converted from one form into another.
- Data transmission: This is a function where data is forwarded to the place where it would be used further.

Metadata (meta data, or sometimes meta information) is "data about data", of any sort in any media. An item of metadata may describe a collection of data including multiple content items and hierarchical levels, for example a database schema. In data processing, metadata is definitional data that provides information about or documentation of other data managed within an application or environment. The term should be used with caution as all data is about something, and is therefore metadata.

Database

- Database may be defined in simple terms as a collection of data
- A database is a collection of related data.
- The database can be of any size and of varying complexity.
- A database may be generated and maintained manually or it may be computerized.

Database Management System

- A Database Management System (DBMS) is a collection of program that enables user to create and maintain a database.
- The DBMS is hence a general purpose software system that facilitates the process of defining constructing and manipulating database for various applications.

1.1 Characteristics of DBMS

- To incorporate the requirements of the organization, system should be designed for easy maintenance.
- Information systems should allow interactive access to data to obtain new information without writing fresh programs.
- System should be designed to co-relate different data to meet new requirements.
- An independent central repository, which gives information and meaning of available data is required.
- Integrated database will help in understanding the inter-relationships between data stored in different applications.
- The stored data should be made available for access by different users simultaneously.
- Automatic recovery feature has to be provided to overcome the problems with processing system failure.

DBMS Utilities

- A data loading utility:
Which allows easy loading of data from the external format without writing programs.
- A backup utility:
Which allows to make copies of the database periodically to help in cases of crashes and disasters.
- Recovery utility:
Which allows to reconstruct the correct state of database from the backup and history of transactions.
- Monitoring tools:
Which monitors the performance so that internal schema can be changed and database access can be optimized.

- File organization:

Which allows restructuring the data from one type to another?

1.2 Difference between File system & DBMS

File System

1. File system is a collection of data. Any management with the file system, user has to write the procedures
2. File system gives the details of the data representation and Storage of data.
3. In File system storing and retrieving of data cannot be done efficiently.
4. Concurrent access to the data in the file system has many problems like
 - a. Reading the file while other deleting some information, updating some information
5. File system doesn't provide crash recovery mechanism.

Eg. While we are entering some data into the file if System crashes then content of the file is lost.

6. Protecting a file under file system is very difficult.

DBMS

1. DBMS is a collection of data and user is not required to write the procedures for managing the database.
2. DBMS provides an abstract view of data that hides the details.
3. DBMS is efficient to use since there are wide varieties of sophisticated techniques to store and retrieve the data.
4. DBMS takes care of Concurrent access using some form of locking.
5. DBMS has crash recovery mechanism, DBMS protects user from the effects of system failures.
6. DBMS has a good protection mechanism.

DBMS = Database Management System

RDBMS = Relational Database Management System

A database management system is, well, a system used to manage databases. A relational database management system is a database management system used to manage relational databases. A relational database is one where tables of data can have relationships based on primary and foreign keys.

1.3 Advantages of DBMS.

Due to its centralized nature, the database system can overcome the disadvantages of the file system-based system

1. **Data independency:**

Application program should not be exposed to details of data representation and storage

DBMS provides the abstract view that hides these details.

2. **Efficient data access.:**

DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently.

3. **Data integrity and security:**

Data is accessed through DBMS, it can enforce integrity constraints.

E.g.: Inserting salary information for an employee.

4. **Data Administration:**

When users share data, centralizing the data is an important task, Experience professionals can minimize data redundancy and perform fine tuning which reduces retrieval time.

5. **Concurrent access and Crash recovery:**

DBMS schedules concurrent access to the data. DBMS protects user from the effects of system failure.

6. **Reduced application development time.**

DBMS supports important functions that are common to many applications.

1.4 Functions of DBMS

- **Data Definition:** The DBMS provides functions to define the structure of the data in the application. These include defining and modifying the record structure, the type and size of fields and the various constraints to be satisfied by the data in each field.
- **Data Manipulation:** Once the data structure is defined, data needs to be inserted, modified or deleted. These functions which perform these operations are part of DBMS. These functions can handle plashud and unplashud data manipulation needs. Plashud queries are those which form part of the application. Unplashud queries are ad-hoc queries which performed on a need basis.
- **Data Security & Integrity:** The DBMS contains modules which handle the security and integrity of data in the application.
- **Data Recovery and Concurrency:** Recovery of the data after system failure and concurrent access of records by multiple users is also handled by DBMS.
- **Data Dictionary Maintenance:** Maintaining the data dictionary which contains the data definition of the application is also one of the functions of DBMS.
- **Performance:** Optimizing the performance of the queries is one of the important functions of DBMS.

1.5 Role of Database Administrator.

Typically there are three types of users for a DBMS:

1. The END User who uses the application. Ultimately he is the one who actually puts the data into the system into use in business. This user need not know anything about the organization of data in the physical level.
2. The Application Programmer who develops the application programs. He/She has more knowledge about the data and its structure. He/she can manipulate the data using his/her programs. He/she also need not have access and knowledge of the complete data in the system.
3. The Data base Administrator (DBA) who is like the super-user of the system.

Database Management System

The role of DBA is very important and is defined by the following functions.

- Defining the schema: The DBA defines the schema which contains the structure of the data in the application. The DBA determines what data needs to be present in the system and how this data has to be presented and organized.
- Liaising with users: The DBA needs to interact continuously with the users to understand the data in the system and its use.
- Defining Security & Integrity checks: The DBA finds about the access restrictions to be defined and defines security checks accordingly. Data Integrity checks are defined by the DBA.
- Defining Backup/Recovery Procedures: The DBA also defines procedures for backup and recovery. Defining backup procedure includes specifying what data is to be backed up, the periodicity of taking backups and also the medium and storage place to backup data.
- Monitoring performance: The DBA has to continuously monitor the performance of the queries and take the measures to optimize all the queries in the application.

1.6 Simplified Database System

Environment

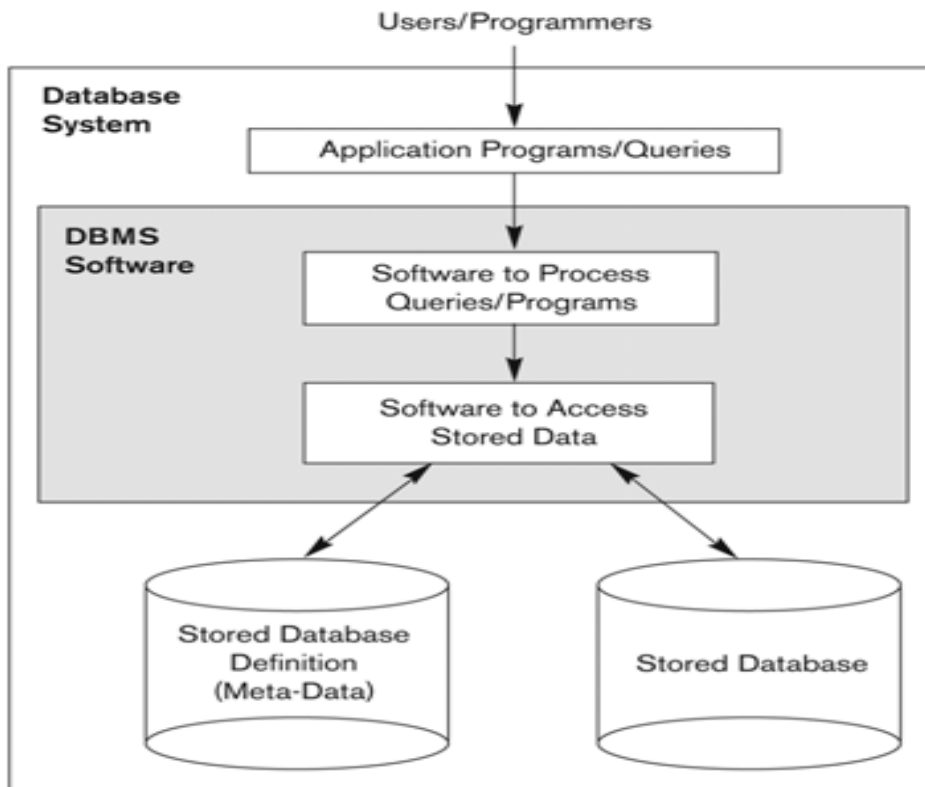


Figure 1.1
A simplified database system environment.

A database management system (DBMS) is a collection of programs that enables users to create and maintain database. The DBMS is a general purpose software system that facilitates the process of defining, constructing, manipulating and sharing databases among various users and applications. Defining a database specifying the database involves specifying the data types, constraints and structures of the data to be stored in the database. The descriptive information is also stored in the database in the form database catalog or dictionary; it is called meta-data.

Manipulating the data includes the querying the database to retrieve the specific data.

An application program accesses the database by sending the queries or requests for data to DBMS.

The important function provided by the DBMS includes protecting the database and maintain the database.

1.7 Example of a Database (with a Conceptual Data Model)

- **Mini-world for the example:**

Part of a UNIVERSITY environment.

- **Some mini-world *entities*:**

STUDENTs

COURSEs

SECTIONs (of COURSEs)

(academic) DEPARTMENTs

INSTRUCTORs

Example of a Database (with a Conceptual Data Model)

- **Some mini-world *relationships*:**

SECTIONs *are of specific* COURSEs

STUDENTs *take* SECTIONs

COURSEs *have prerequisite* COURSEs

INSTRUCTORs *teach* SECTIONs

COURSEs *are offered by* DEPARTMENTs

STUDENTs *major in* DEPARTMENTs

Example of a simple Database

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

Example of a simple Database

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

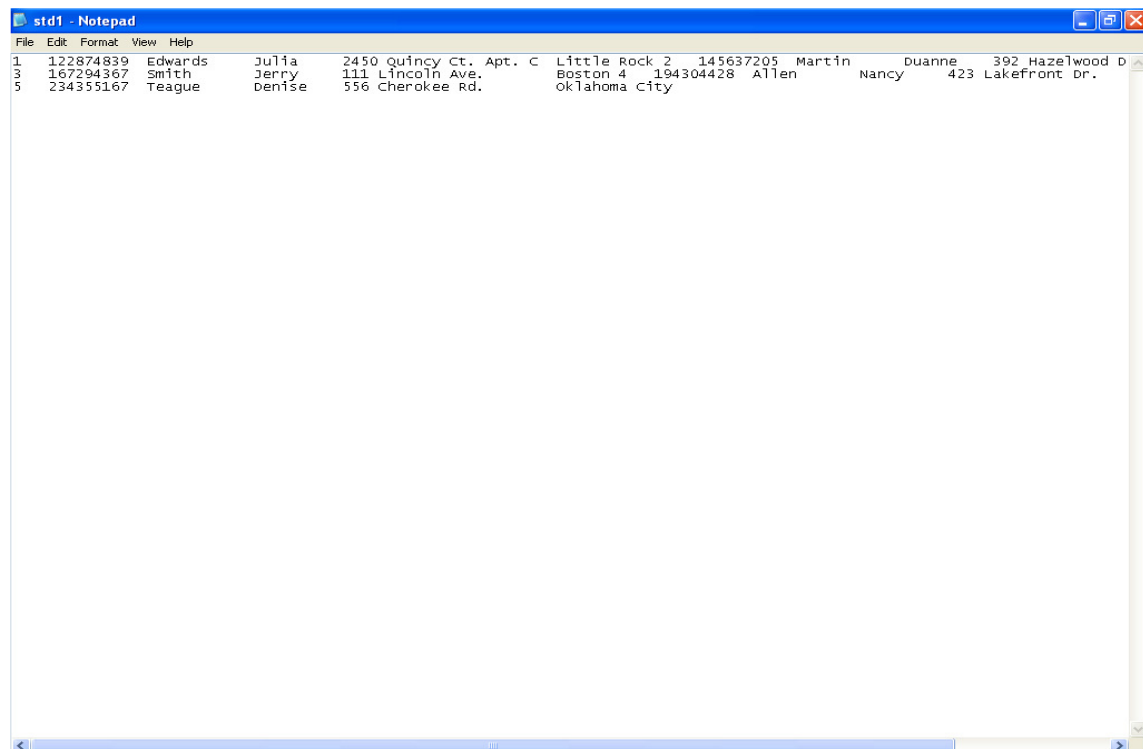
PREREQUISITE

Course_number	Prerequisite_number
C-S3380	C-S3320
C-S3380	MATH2410
C-S3320	CS1310

Figure 1.2

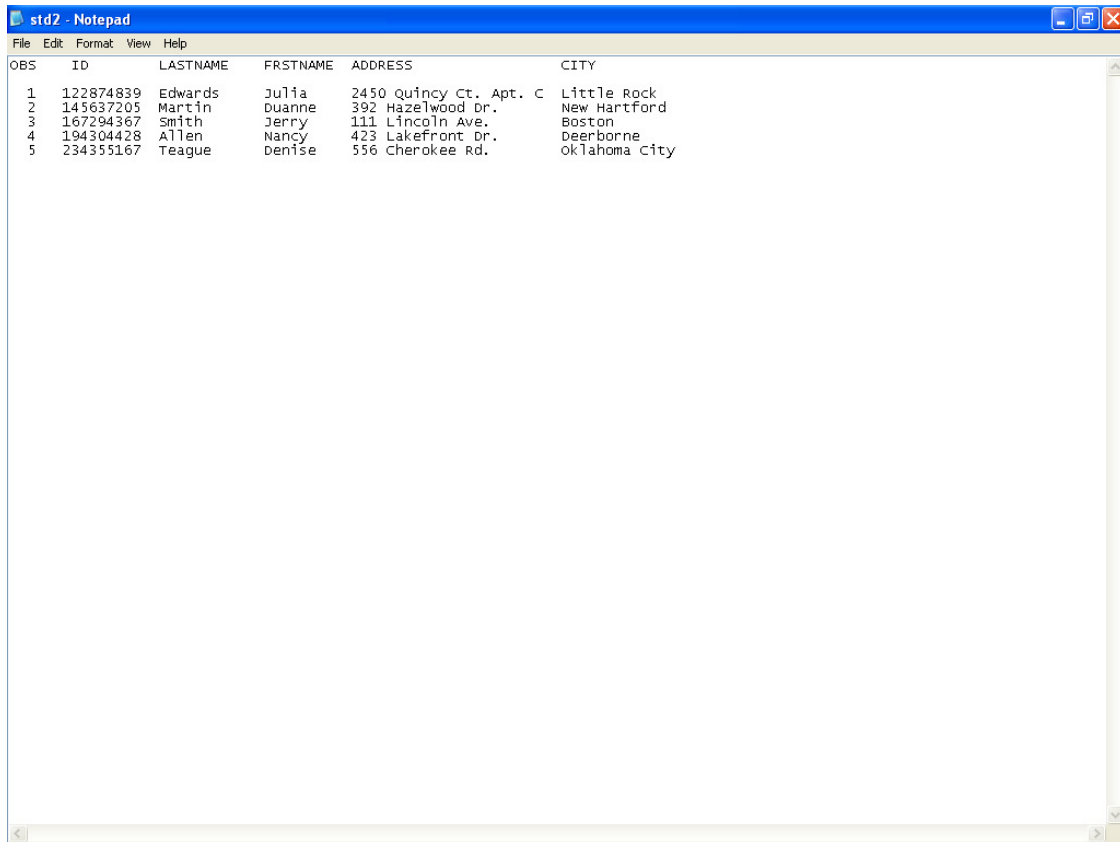
A database that stores student and course information.

Example of a Student File



ID	Name	Address	Phone Number
122874839	Edwards	Julia 2450 quincy Ct. Apt. C	Little Rock 2 145637205
167294367	Smith	Jerry 111 Lincoln Ave.	Boston 4 194304428
234355167	Teague	Denise 556 Cherokee Rd.	oklahoma city
			Martin Duanne 392 Hazelwood Dr
			Allen Nancy 423 Lakefront Dr.

Example of a Student File



The screenshot shows a Notepad window titled "std2 - Notepad" with a menu bar (File, Edit, Format, View, Help). The text inside is a student file with the following data:

OBS	ID	LASTNAME	FRSTNAME	ADDRESS	CITY
1	122874839	Edwards	Julia	2450 Quincy Ct. Apt. C	Little Rock
2	145637205	Martin	Duanne	392 Hazelwood Dr.	New Hartford
3	167294367	Smith	Jerry	111 Lincoln Ave.	Boston
4	194304428	Allen	Nancy	423 Lakefront Dr.	Deerborne
5	234355167	Teague	Denise	556 Cherokee Rd.	oklahoma city

Example of a simplified database catalog

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

Figure 1.3

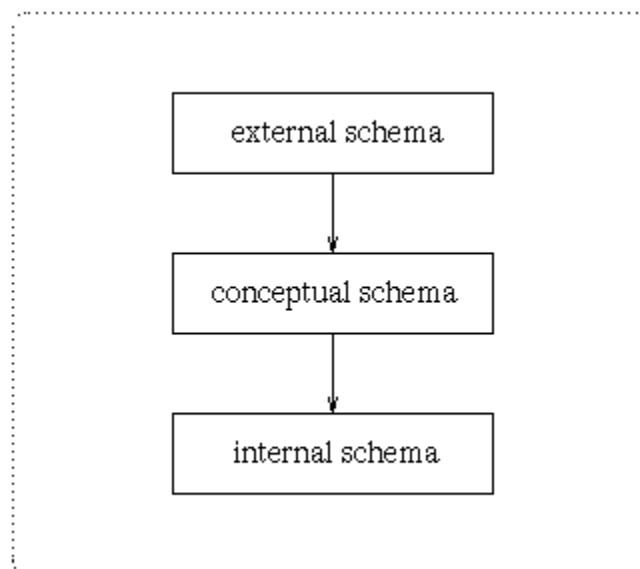
An example of a database catalog for the database in Figure 1.2.

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....
....
....
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerated type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

1.8 Architecture of DBMS



A commonly used views of data approach is the three-level architecture suggested by ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee). ANSI/SPARC produced an interim report in 1972 followed by a final report in 1977. The reports proposed an architectural framework for databases. Under this approach, a database is considered as containing data about an *enterprise*. The three levels of the architecture are three different views of the data:

External - individual user view

Conceptual - community user view

Internal - physical or storage view

The three level database architecture allows a clear separation of the information meaning (conceptual view) from the external data representation and from the physical data structure layout. A database system that is able to separate the three different views of data is likely to be flexible and adaptable. This flexibility and adaptability is data independence that we have discussed earlier.

We now briefly discuss the three different views.

The external level is the view that the individual user of the database has. This view is often a restricted view of the database and the same database may provide a number of different views for different classes of users. In general, the end users and even the application programmers are only interested in a subset of the database. For example, a department head may only be interested in the departmental finances and student enrolments but not the library information. The librarian would not be expected to have any interest in the information about academic staff. The payroll office would have no interest in student enrolments.

The conceptual view is the information model of the enterprise and contains the view of the whole enterprise without any concern for the physical implementation. This view is normally more stable than the other two views. In a database, it may be desirable to change the internal view to improve performance while there has been no change in the

conceptual view of the database. The conceptual view is the overall community view of the database and it includes all the information that is going to be represented in the database. The conceptual view is defined by the conceptual schema which includes definitions of each of the various types of data.

The internal view is the view about the actual physical storage of data. It tells us what data is stored in the database and how. At least the following aspects are considered at this level:

Storage allocation e.g. B-trees, hashing etc.

Access paths e.g. specification of primary and secondary keys, indexes and pointers and sequencing.

Miscellaneous e.g. data compression and encryption techniques, optimization of the internal structures.

Efficiency considerations are the most important at this level and the data structures are chosen to provide an efficient database. The internal view does not deal with the physical devices directly. Instead it views a physical device as a collection of physical pages and allocates space in terms of logical pages.

The separation of the conceptual view from the internal view enables us to provide a logical description of the database without the need to specify physical structures. This is often called *physical data independence*. Separating the external views from the conceptual view enables us to change the conceptual view without affecting the external views. This separation is sometimes called *logical data independence*.

Assuming the three level view of the database, a number of mappings are needed to enable the users working with one of the external views. For example, the payroll office may have an external view of the database that consists of the following information only:

Staff number, name and address.

Staff tax information e.g. number of dependents.

Staff bank information where salary is deposited.

Staff employment status, salary level, leave information etc.

The conceptual view of the database may contain academic staff, general staff, casual staff etc. A mapping will need to be created where all the staff in the different categories are combined into one category for the payroll office. The conceptual view would include information about each staff's position, the date employment started, full-time or part-time etc. This will need to be mapped to the salary level for the salary office. Also, if there is some change in the conceptual view, the external view can stay the same if the mapping is changed.

1.9 Data Independence

Data independence can be defined as the capacity to change the schema at one level without changing the schema at next higher level. There are two types of data Independence. They are

1. Logical data independence.
 2. Physical data independence.
-
1. Logical data independence is the capacity to change the conceptual schema without having to change the external schema.
 2. Physical data independence is the capacity to change the internal schema without changing the conceptual schema.

When not to use a DBMS

- Main inhibitors (costs) of using a DBMS:
 - High initial investment and possible need for additional hardware.
 - Overhead for providing generality, security, concurrency control, recovery, and integrity functions
- When a DBMS may be unnecessary:

- If the database and applications are simple, well defined and not expected to change.
- If there are stringent real-time requirements that may not be met because of DBMS overhead.
- If access to data by multiple users is not required.
- When no DBMS may suffice:
- If the database system is not able to handle the complexity of data because of modeling limitations
- If the database users need special operations not supported by the DBMS.

1.10 Types of Databases and Database Applications

- Traditional Applications:
Numeric and Textual Databases
- More Recent Applications:
Multimedia Databases
Geographic Information Systems (GIS)
Data Warehouses
Real-time and Active Databases
Many other applications

1.11 Data Model

A model is an abstraction process that hides superfluous details. Data modeling is used for representing entities of interest and their relationship in the database.

Data model and different types of Data Model

Data model is a collection of concepts that can be used to describe the structure of a database which provides the necessary means to achieve the abstraction. The structure of a database means that holds the data.

→ data types

→ relationships

→ constraints

Types of Data Models

1. High Level- Conceptual data model.
2. Low Level – Physical data model.
3. Relational or Representational
4. Object-oriented Data Models:
5. Object-Relational Models:

1. High Level-conceptual data model: User level data model is the high level or conceptual model. This provides concepts that are close to the way that many users perceive data.

2 .Low level-Physical data model : provides concepts that describe the details of how data is stored in the computer model. Low level data model is only for Computer specialists not for end-user.

3. Representation data model: It is between High level & Low level data model

Which provides concepts that may be understood by end-user but that are not too far removed from the way data is organized by within the computer.

The most common data models are

1. Relational Model

The Relational Model uses a collection of tables both data and the relationship among those data. Each table have multiple column and each column has a unique name .

Relational database comprising of two tables

Customer –Table.

Customer-Name	Security Number	Address	City	Account-Number
Preethi	111-222-3456	Yelhanka	Bangalore	A-101
Sharan	111-222-3457	Hebbal	Bangalore	A-125
Preethi	112-123-9878	Jaynagar	Bangalore	A-456
Arun	123-987-9909	MG road	Bangalore	A-987
Preethi	111-222-3456	Yelhanka	Bangalore	A-111
Rocky	222-232-0987	Sanjay Nagar	Bangalore	A-111

Account –Table

Account-Number	Balance
A-101	1000.00
A-125	1200.00
A-456	5000.00
A-987	1234.00
A-111	3000.00

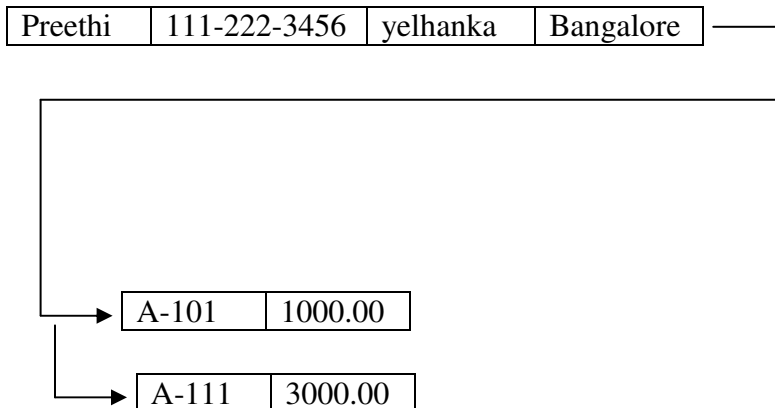
Customer Preethi and Rocky share the same account number A-111

Advantages

1. The main advantage of this model is its ability to represent data in a simplified format.
2. The process of manipulating record is simplified with the use of certain key attributes used to retrieve data.
3. Representation of different types of relationship is possible with this model.

2. Network Model

The data in the network model are represented by collection of records and relationships among data are represented by links, which can be viewed as pointers.



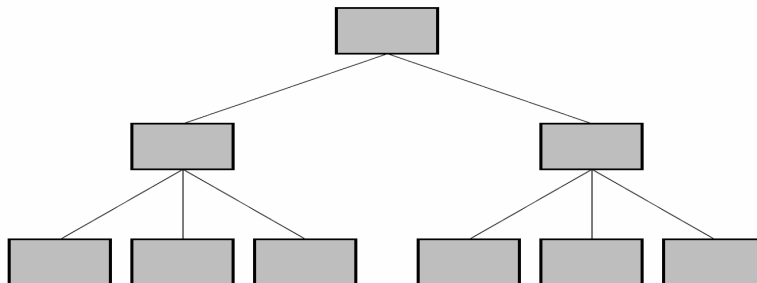
The records in the database are organized as collection of arbitrary groups.

Advantages:

1. Representation of relationship between entities is implemented using pointers which allows the representation of arbitrary relationship
2. Unlike the hierarchical model it is easy.
3. data manipulation can be done easily with this model.

3. Hierarchical Model

A hierarchical data model is a data model which the data is organized into a tree like structure. The structure allows repeating information using parent/child relationships: each parent can have many children but each child only has one parent. All attributes of a specific record are listed under an entity type.



Advantages:

1. The representation of records is done using an ordered tree, which is natural method of implementation of one-to-many relationships.
2. Proper ordering of the tree results in easier and faster retrieval of records.
3. Allows the use of virtual records. This result in a stable database especially when modification of the data base is made.

4.0 Object-oriented Data Models

- Several models have been proposed for implementing in a database system.
- One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE).
- Additionally, systems like O2, ORION (at MCC – then ITASCA), IRIS (at H.P.- used in Open OODB).

5.0 Object-Relational Models

- Most Recent Trend. Started with Informix
- Universal Server.
- Relational systems incorporate concepts from object databases leading to object-relational.
- Object Database Standard: ODMG-93, ODMG-version 2.0, ODMG-version 3.0.
- Exemplified in the latest versions of Oracle-10i, DB2, and SQL Server and other DBMSs.
- Standards included in SQL-99 and expected to be enhanced in future SQL standards.

Schemas versus Instances

- 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 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.

DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- High-Level or Non-procedural Languages: These include the relational language SQL
- May be used in a standalone way or may be embedded in a programming language
- Low Level or Procedural Languages:

These must be embedded in a programming language

Data Definition Language (DDL)

Used by the DBA and database designers to specify the conceptual schema of a database.

- In many DBMSs, the DDL is also used to define internal and external schemas (views).

- In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define **internal and external** schemas.
- SDL is typically realized via DBMS commands provided to the DBA and database designers

Data Manipulation Language (DML)

Used to specify database retrievals and updates DML commands (data sublanguage) can be *embedded in a general-purpose programming language* (host language), such as COBOL, C, C++, or Java.

- A library of functions can also be provided to access the DBMS from a programming language
- Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

Types of DML

- **High Level or Non-procedural Language:**

For example, the SQL relational language are “set”-oriented and specify what data to retrieve rather than how to retrieve it.

Also called **declarative languages**.

- **Low Level or Procedural Language:**

Retrieve data one record-at-a-time;

Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

DBMS Interfaces

- Stand-alone query language interfaces
Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)

- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
- Menu-based, forms-based, graphics-based, etc.

DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
- **Embedded Approach: e.g. embedded SQL (for C,C++, etc.), SQLJ (for Java)**
- **Procedure Call Approach: e.g. JDBC for Java, ODBC for other programming languages**
- **Database Programming Language Approach:**
e.g. ORACLE has PL/SQL, a programming language based on SQL;
language incorporates SQL and its data types as integral components/

User-Friendly DBMS Interfaces

- Menu-based, popular for browsing on the web
- Forms-based, designed for naïve users
- Graphics-based (Point and Click, Drag and Drop, etc.)
- Natural language: requests in written English
- Combinations of the above: For example, both menus and forms used extensively in Web database interfaces

Other DBMS Interfaces

- Speech as Input and Output
- Web Browser as an interface
- Parametric interfaces, e.g., bank tellers using function keys.
- Interfaces for the DBA:
 - Creating user accounts, granting authorizations
 - Setting system parameters
 - Changing schemas or access paths

2.0 The database system environment

The DBMS is a complex software system.

Typical DBMS Component Modules

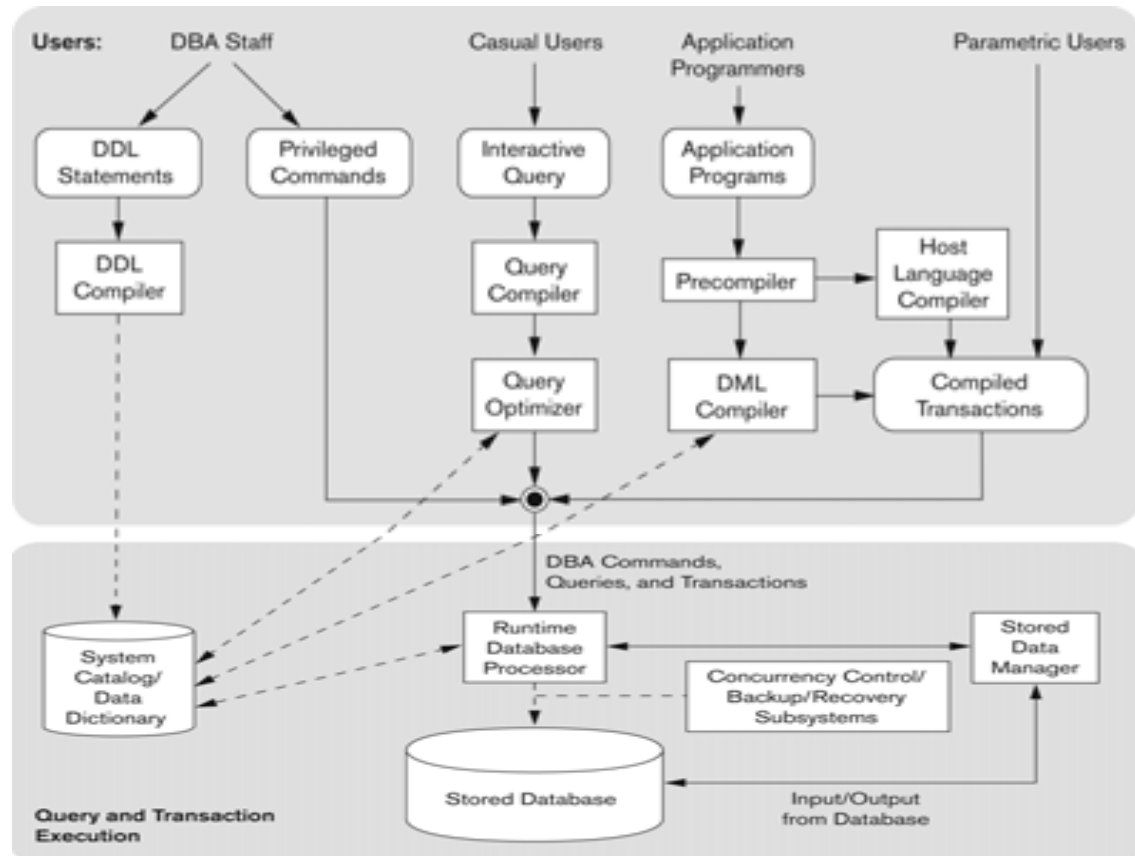


Figure 2.3
Component modules of a DBMS and their interactions.

The figure is divided into two halves. The top half of the figure refers to the various users of the database environment and their interfaces. The lower half shows the internals of the DBMS responsible for storage of data and processing of transaction.

The database and the DBMS catalog are usually stored on disk. Access to the disk is primarily controlled by operating system(OS).which includes disk input/Output. A higher level stored data manager module of DBMS controls access to DBMS information that is stored on the disk.

If we consider the top half of the figure, It shows interfaces to DBA staff, casual users, application programmers and parametric users

The DDL compiler processes schema definitions, specified in the DDL, and stores the description of the schema in the DBMS Catalog. The catalog includes information such as names and sizes of the files, data types of data items. Storage details of each file, mapping information among schemas and constraints.

Casual users and persons with occasional need of information from database interact using some form of interface which is interactive query interface. The queries are parsed, analysed for correctness of the operations for

the model. the names of the data elements and so on by a query compiler that compiles them into internal form. The internal query is subjected to query optimization. The query optimizer is concerned with rearrangement and possible recording of operations, eliminations of redundancies.

Application programmer writes programs in host languages. The precompiler extracts DML commands from an application program

2.1 Centralized and Client-Server DBMS Architectures

Centralized DBMS:

- Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
- User can still connect through a remote terminal – however, all processing is done at centralized site.

A Physical Centralized Architecture

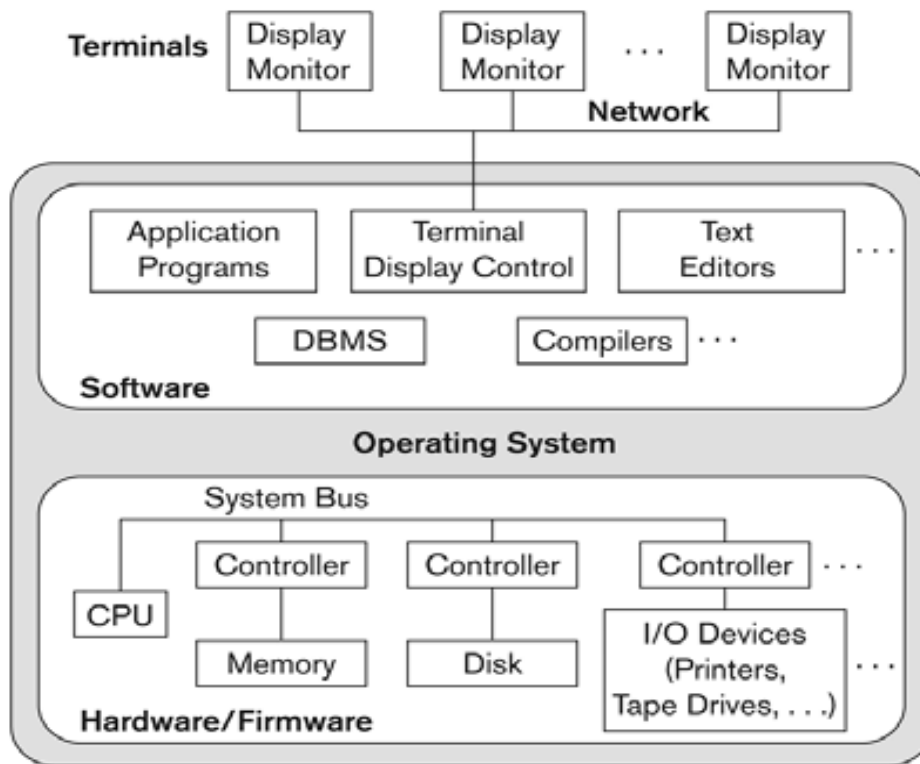


Figure 2.4
A physical centralized architecture.

Architectures for DBMS have followed trends similar to those generating computer system architectures. Earlier architectures used mainframes computers to provide the main processing for all system functions, including user application programs and user interface programs as well all DBMS functionality. The reason was that most users accessed such systems via computer terminals that did not have processing power and only provided display capabilities. Therefore all processing was performed remotely on the computer system, and only display information and controls were sent from the computer to the display terminals, which were connected to central computer via various types of communication networks.

As prices of hardware declined, most users replaced their terminals with PCs and workstations. At first database systems used these computers similarly to how they have used is play terminals, so that 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.

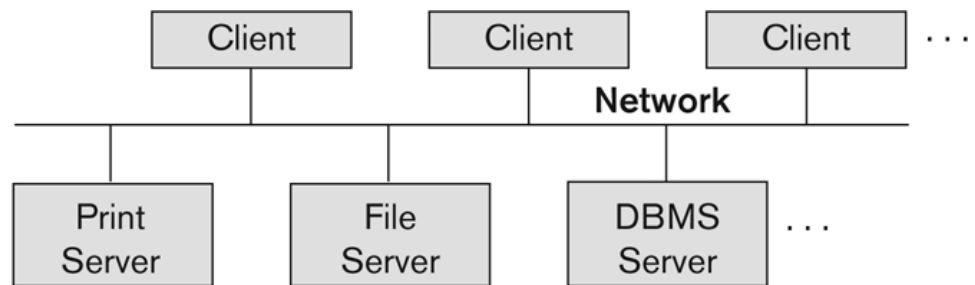
Basic 2-tier Client-Server Architectures

- Specialized Servers with Specialized functions
- Print server
- File server
- DBMS server
- Web server
- Email server
- Clients can access the specialized servers as needed

Logical two-tier client server architecture

Figure 2.5

Logical two-tier client/server architecture.



Clients

- Provide appropriate interfaces through a client software module to access and utilize the various server resources.
- Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.
- Connected to the servers via some form of a network.
- (LAN: local area network, wireless network, etc.)

DBMS Server

- Provides database query and transaction services to the clients
- Relational DBMS servers are often called SQL servers, query servers, or transaction servers
- Applications running on clients utilize an Application Program Interface (API) to access server databases via standard interface such as:

- ODBC: Open Database Connectivity standard
- JDBC: for Java programming access
- Client and server must install appropriate client module and server module software for ODBC or JDBC

Two Tier Client-Server Architecture

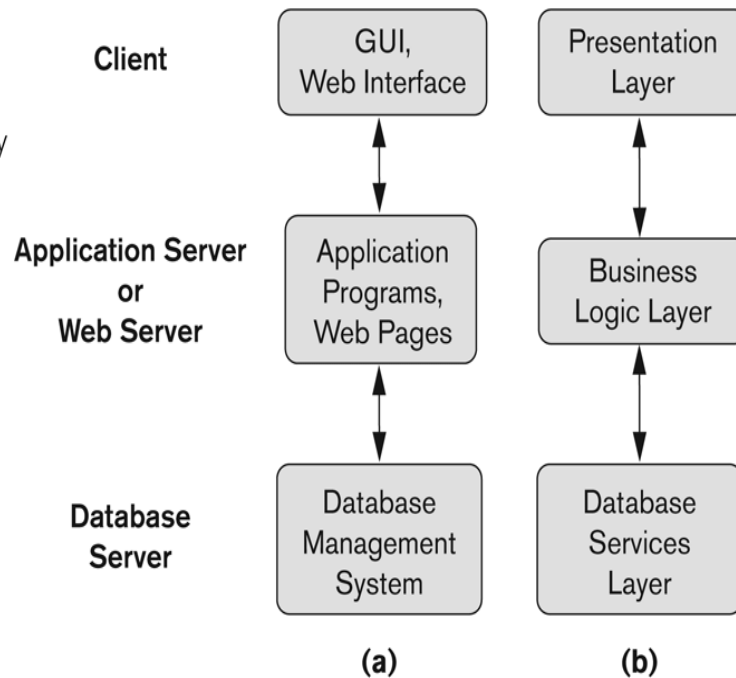
- A client program may connect to several DBMSs, sometimes called the data sources.
- In general, data sources can be files or other non-DBMS software that manages data. Other variations of clients are possible: e.g., in some object DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc.

Three Tier Client-Server Architecture

- Common for Web applications
- Intermediate Layer called Application Server or Web Server:
- Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
- Acts like a conduit for sending partially processed data between the database server and the client.
- Three-tier Architecture Can Enhance Security:
- Database server only accessible via middle tier
- Clients cannot directly access database server

Figure 2.7

Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.



Classification of DBMSs

- Based on the data model used
- Traditional: Relational, Network, Hierarchical.
- Emerging: Object-oriented, Object-relational.
- Other classifications
- Single-user (typically used with personal computers) vs. multi-user (most DBMSs).
- Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

Variations of Distributed DBMSs (DDBMSs)

- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated or Multidatabase Systems
- Distributed Database Systems have now come to be known as client-server based database systems because:

- They do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.

Cost considerations for DBMSs

- Cost Range: from free open-source systems to configurations costing millions of dollars
- Examples of free relational DBMSs: MySQL, PostgreSQL, others