Module-1

Introduction to Databases: Introduction, An Example, Characteristics of Database approach, Advantages of using DBMS approach, When not to use a DBMS

Database System Concepts and Architecture: Data models, Schemas and instances, Three schema architecture and data independence Database languages and interfaces, The database system environment,

SQL: SQL Data Definition and Data Types specifying basic constraints in SQL, Basic retrieval queries in SQL, Insert, Delete and Update statements in SQL, Additional features of SQL, More complex SQL Queries, Specifying Constraints as Assertions and Triggers, Views (Virtual Tables) in SQL, Schema Change Statement in SQL.

Text Book:

1. Database systems Models, Languages, Design and Application Programming, RamezElmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.

2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill

Prepared by,
Dr.Aruna M G
Associate Professor
Department of AI&ML
DSCE
Bangalore

Introduction to Database System

Different Applications of Traditional Database

- **Traditional Database applications**: The information stored and accessed in computer either numeric or textual pattern. Such type of databases is called Traditional Database applications. The applications are classified as follows.

  1. **Multimedia Databases**: Store picture, video clips and sound message.
  2. **Geographic Information Systems (GIS)**: It can store and analysis maps, weather data and satellite images.
  3. **Data Warehouses and online analytical processing (OLAP)**: This is used for decision making
  4. **Real-time and Active Databases**: It can be used in controlling industrial and manufacturing processes.
  5. **Database Search Technique:** Used in World Wide Web (www) to improve the search for information that is needed for browsing in the internet.

  Database plays a critical role in almost all areas. Where computers are used such as business, engineering, medicine, law, education and library science etc.

<u>Define the following</u>:

Data

Data means recorded facts or Known facts that can be recorded and that have implicit (unquestioned) meaning.

Ex: Roll No, Name, Telephone Number and Address of the people will be recorded in indexed address book, hard drive and S/w such as MS-Access or Ms-Excel. This collection of related data an implicit meaning and hence it is database.

Database

A database is a collection of related data or information, which stores data in form of table.

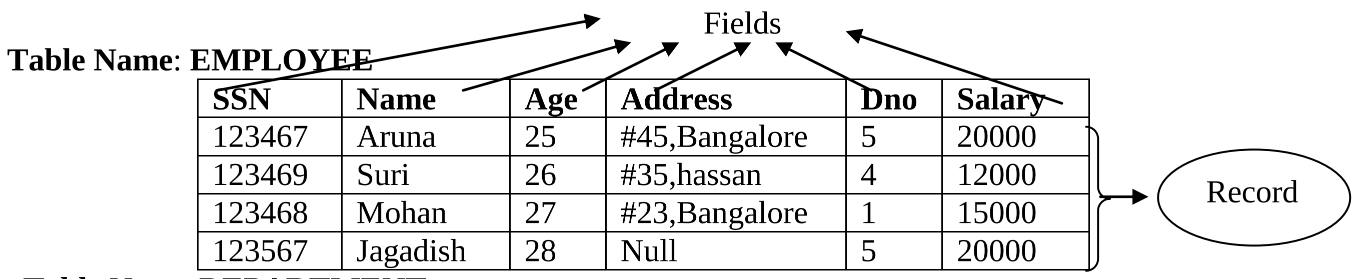


Table Name: **EMPLOYEE**

SSN	Name	Age	Address	Dno	Salary
123467	Aruna	25	#45,Bangalore	5	20000
123469	Suri	26	#35,hassan	4	12000
123468	Mohan	27	#23,Bangalore	1	15000
123567	Jagadish	28	Null	5	20000

Table Name:**DEPARTMENT**

Dname	Dnumber	Dlocation
Research	5	Bangalore
Administration	4	Bangalore
Headquarters	1	Bombay

Fig 1.1:A database that stores employee and department information

Database System

A database system is a software system, which supports the definition and use of a database
DDL: Data Definition Language
DML: Data Manipulation Language

Database Management System (DBMS)

A DBMS is a collection of interrelated files and set of programs that allow users to access and modify these files.
or
A DBMS is a collection of programs that enables the users to create & maintain a database. DBMS is general-purpose software that facilitates the process of defining, constructing, manipulating and sharing database among various users and applications.

- ❖ **Defining a Database:** It involves the specifying types of data types, structures and constraints of the data to be stored in the database.

- ❖ **Constructing a Database:** The process of storing the database on a secondary storage medium.

- ❖ **Manipulating a Database:** It includes functions for querying (retrieve specific data), updating, reporting, insertions, deletions and modifications to its content.

- ❖ Concurrent processing and sharing by a set of users and programs – yet, keeping all data valid and consistent.

Other features:

- ❖ **Protecting the Database**: It involves the two types of protection i.e. system protection against hardware/software (crashes) and security protection against unauthorized access.

- ❖ **Maintaining the Database**: Maintain the database over long period of time.

Example's: Access, Oracle, MySQLServer, dBase, Sybase, Clipper, etc.

<u>File Processing System versus a DBMS</u>

File processing system was the earliest storage type database system. In file processing system the permanent records are stored in various files, and different application programs are written to extract records from, and to add records to appropriate files. To keep organizational information in a file processing system has some disadvantage

- ➢ **Data redundancy**: The same data stored in multiple places. This leads uncontrolled redundancy in file system. This redundancy leads higher storage and access cost.
  Example: Two files of Banking system have savings-account records and a checking-account record contains the two common fields i.e. address and telephone of customer appear in both files.

CUSTOMER_SAVING

Account_number	Name	Age	Address
123467	Aruna	25	#45,Bangalore
123469	Suri	26	#35,hassan

CHECKING_ACCOUNT

Account_number	Name	Address
123467	Aruna	#45,Bangalore
123469	Suri	#35,hassan
123468	Mohan	#23,Bangalore

Figure1.2 Data redundancy

- **Inconsistency data**: Incorrectness in data entry.
  Example: A changes made in customer address in savings-account records will be reflected but not in other files of banking system this makes the data inconsistency.

CUSTOMER_SAVING

Account_number	Name	Age	Balance	Address
123467	Aruna	25	20000	#45,Bangalore
123469	Suri	26	12000	#35,hassan
123468	Mohan	27	15000	#23,Bangalore
123567	Jagadish	28	20000	#323,Bangalore
123467	Arun	25	20000	#45,Bangalore

Spelling Mistake in name field

CHECKING_ACCOUNT

Account_number	Name	Address
123467	Aruna	#45,Bangalore
123469	Suri	#35,hassan
123468	Mohan	#23,Bangalore
123567	Jagadish	Null

Changes not reflected

Figure1.3 Data Inconsistency

- **Inflexibility**: If any necessary changes like adding a file or extend the data elements in an existing file the entire file has to be changed. This makes the affecting the stored data and the existing applications programs.

- **Difficulty in accessing data**: If requested query is not designed for original system. Then check the particular file to get the requested information manually or ask the system programmer to write the needed application program.
  Example: If bank officer's needs to find out the names of all customers who live within a particular postal-code area in a city's. There is no application program on hand to meet it in the original system.

The file-processing systems do not allow needed data to be retrieved in a convenient and efficient manner.

- **Data isolation:** The data are scattered in various files, and files may be different formats, it is difficult to write new application programs to retrieve the appropriate data.

- **Integrity problems**: The data values stored in the database must satisfy certain types of consistency constraints.
  Example: The balance of a bank account may never fall below Rs.100. Developers enforce these constraints in the system by adding code in the various application programs. If new constraints wanted to add, it is difficult to change the programs.

- **Atomicity problems:** If any mechanical or electrical types of failure occur to the system, the data should be restored to the consistent state. Atomic means it must happen entirely or not at all.
  Example: If J wants to transfer Rs2000 from his account to S. If a system failure occur during execution of the program, that Rs2000 is removed from J but not credited to account S, resulting in an inconsistent state. To stay in consistent state either both credit and debit occur, or that neither occur.

CUSTOMER_SAVING

Account_number	Name	Age	Balance	Address
123469	Suri	26	12000	#35,hassan
123468	Mohan	27	15000	#23,Bangalore
123567	Jagadish	28	18000	#323,Bangalore

Figure1.4 Atomicity problems inconsistent state

- **Concurrent access anomalies:** System allows multiple users to update, delete and retrieve the data simultaneously. If two programs run concurrently, they may both read the value, and write back the value. Depending on which one writes the value last, that value maybe correct or incorrect value.
  Example**:** If two customers withdraw amount from J (say Rs200 and Rs300 respectively) at the same time, the result of concurrent executions may leave the account in an incorrect state.

- **Security problem:** Not every user of the database system should be able to access all the data.
  Example: In a student database system, updating marks is allowed only for lecturer, only viewing the data for student and restricted for updating the data.

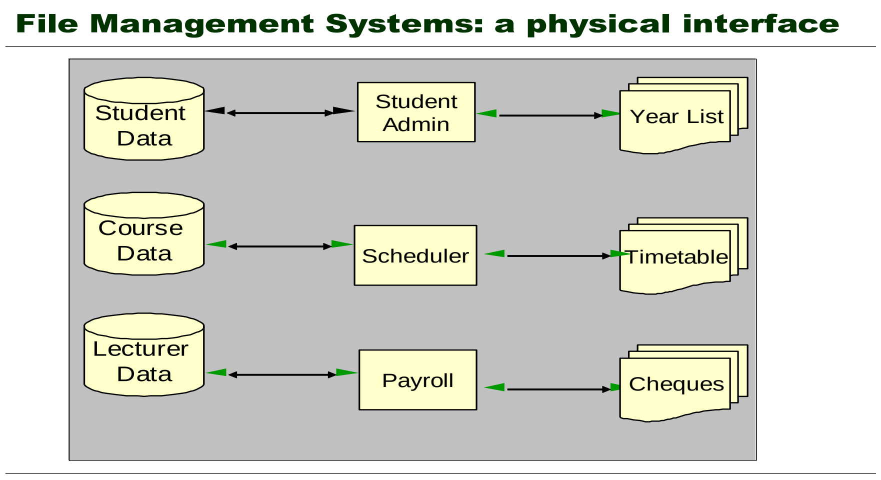
- **Limited data sharing**

- **Poor enforcement of standards**

- **Low programmer productivity**

- **Excessive program maintenance**
- **Excessive data maintenance**

To overcome these drawbacks of file-processing system the DBMS was introduced.



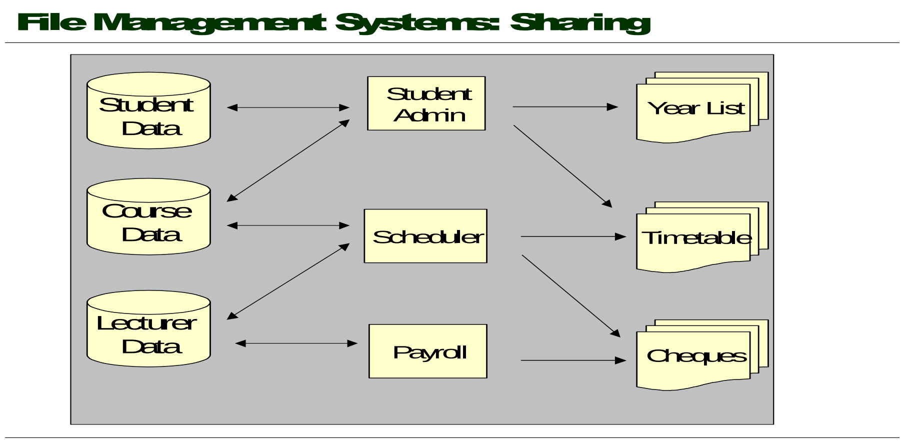


Figure1.5: Storage Representation of File Processing System

DBMS: A Logical Interface

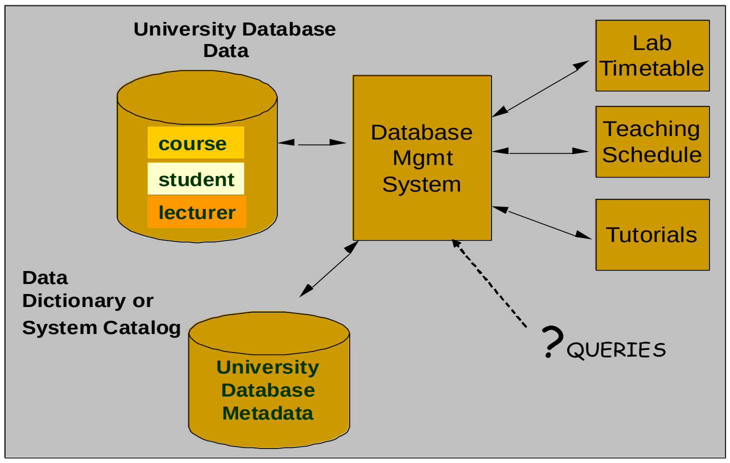


Figure1.6: Storage Representation of Database Management System

Main Characteristics of the Database Approach versus the file processing system

- **Self-describing nature of a database system**: A DBMS contains not only the database but also a definition or description of the database structure and constraints. This definition is stored in the DBMS catalog, which contains information like structure of each file, the type and storage format of each data item, and various constraints on the data. The information stored in the catalog is called **meta-data**.

  For the example shown in figure1.1, the DBMS catalog will store the definitions of all the files shown in figure1.2shows some entries in a database catalog. These definitions are specified by the database designer to creating and storing the database in the catalog.

RELATIONS

Relation_name	No_of_columns
EMPLOYEE	5
DEPARTMENT	3
PROJECT	3
DEPENDENT	4

(a)

COLUMNS

Column_name	Data_type	Belongs_to_relaiton
SSN	Character(10)	EMPLOYEE
Name	Character(10)	EMPLOYEE
Age	Integer	EMPLOYEE
Address	Character(30)	EMPLOYEE
Salary	Decimal(10,2)	EMPLOYEE
Dname	Character(10)	DEPARTMENT
Dnumber	Character(10)	DEPARTMENT
Dlocation	Character(30)	DEPARTMENT
Pname	Character(10)	PROJECT
…..	……..	…..
……	……..	…….

(b)

Figure1.7:An example of a database catalog for the database in figure1.1.

❖ **Insulation between programs and data**: This allows changing data storage structures and operations without having to change the DBMS access programs. This property is called **program-data independence**.

Example: To add a new field Birth_date to Employee records, we only need to change the description of Employee records in the catalog; no change in the programs.

Data Item Name	Starting Position in Record	Length in characters (bytes)
SSN	1	10
Name	11	10
Age	21	1
Address	22	30
Salary	52	10,2

Figure1.8 internal storage for EMPLOYEE record, based on the database catalog in Figure1.1

In object-oriented and object-relational systems, users can define operations on data as part of the database definitions.

An **operation** (also called a function or method) is specified in two parts.

- The interface (or signature) of an operation includes the operation name and the data types of its arguments (or parameters).
- The implementation (or method) of the operation is specified separately and can be changed without affecting the interface.

User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented. This may be termed **program-operation independence**.

❖ **Data Abstraction**: The characteristic that allows program-data independence and program-operation independence is called **data abstraction**. A data model is used to hide storage and implementation details and also present the users with a conceptual view of the database.

❖ **Support of multiple views of the data**: Each user may see a different view of the database, which describes only the data of interest to that user.
Example: One user is interested in accessing and printing the specified employee field details. A second user, who is interested in checking salary details of employee who work in department 5.

(a) EMPLOYEE

SSN	Name
123467	Aruna
123469	Suri
123468	Mohan
123567	Jagadish

(b)EMPLOYEE

SSN	Name	Dno
123467	Aruna	5
123567	Jagadish	5

Figure1.9 Two views derived from database in figure1.1 (a) User1 view
(b) User2 view

❖ **Sharing of data and multiuser transaction processing**: A multiuser DBMS allowing a set of multiple users to retrieve and to update the database at the same time. Concurrency control within the DBMS guarantees that each transaction is correctly executed or completely aborted. OLTP (Online Transaction Processing) is a major part of database applications.

Example: when multiuser accesses the information about the seat details of the flight and reserve a seat on an airline reservation database system ensure that each seat can be accessed by only one user at a time. The DBMS software ensures that concurrent transactions operate correctly and efficiently.

Database User

Users may be divided into those who actually use and control the content (called “Actors on the Scene”) and those who enable the database to be developed and the DBMS software to be designed and implemented (called “Workers Behind the Scene”).

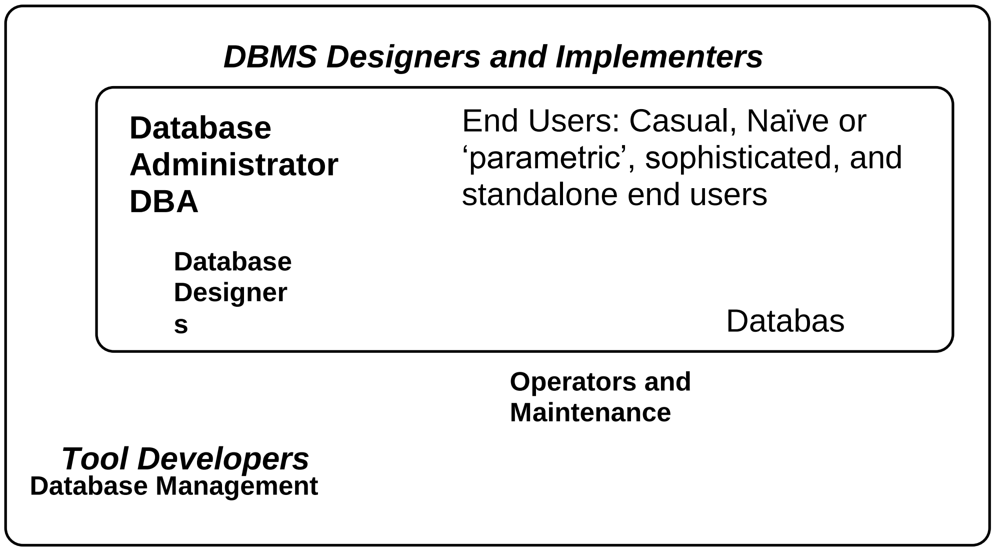

DBMS Designers and Implementers
Database Administrator DBA
End Users: Casual, Naïve or 'parametric', sophisticated, and standalone end users
Database Designers
Databas
Operators and Maintenance
Tool Developers
Database Management

Figure1.10 Users of DBMS

ACTORS ON THE SCENE OR PEOPLE THAT WORK WITH DATABASES

- System Analysts and Application Developers
- Database Designers
- Database Administrators
- End Users

System Analysts and Application programmers
Also known as software engineers or software developers, System analysis determines the requirement of end users (naive end users) and develops specifications for canned transactions that meet these requirements.

Application Programmer: Application programmer are responsible for developing package (or writing program) that facilitates data access for the end users.
Responsible of Application developer

- Implement the database design.
- Implement the application programs to meet the program specifications.
- Test and debug the database implementation and the application programs.
- Document the database implementation and the application programs.

Database administrators: In any organization where many people use the same resources (h/w or s/w like data and programs), there need to have central control over DBMS to manage these resources. The people who manage these resources are called Database administrator. Database administrator is responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software, and hardware resources, controlling its use and monitoring efficiency of operations.
The Responsible of Database Administrator (DBA):

- **Design of the conceptual and physical schemas**: The DBA must design the conceptual schema (decide what relations to store) and the physical schema (decide how to store them). The DBA creates the original database schema, storage structure and access methods by writing a set of definitions that is translated by the data-storage and data-definition-language (DDL) compiler to a set of tables that is stored permanently in the **data dictionary**.

- **Security and Authorization**: The unauthorized user accessing the data in DBMS is not permitted. The DBA is responsible for granting/permissions of different types of authorization to access only certain views and relations for DBMS users.
- **Data Availability and Recovery from Failures:** If the system get fails, the users can continue to access the valid data by the availability of data in DBMS which is not corrupted. The DBA must also work to restore the data to consistent state i.e. any types of failure occur during transaction.
- **Database Tuning**: User's needs should be provided with in time. The DBA is responsible for modifying the DB, in conceptual and physical schemas, to ensure adequate performance as requirements changes.
- **Integrity-constraint Specification**: The data values stored in the DB must satisfy certain consistency constraints. For example, If the number of hours an employee may work in 1 week which may not exceed a specified limit (say, 48hrs). Such constraints must specified explicitly by the DBA. The integrity constraints are kept in a special system structure in DBMS.

Database Designers: Database Designers are responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.
Responsible of Database designer

- Choose appropriate structures to represent the information specified by the system analysts.
- Choose appropriate structures to store the information in a normalized manner in order to guarantee integrity and consistency of data.
- Choose appropriate structures to guarantee an efficient system.
- Document the database design.

End-users: This class of users job; is to access the DB for querying, updating and generating reports. There are several groups of end users:

- **Casual end users**: They access database occasionally, but they need different information each time. They use sophisticated DB query language to specify their requests.

- **Naive or Parametric end users**: They access the DB constantly, with standard types of queries and updates called **canned transactions** that have been carefully programmed and tested, This type of users who interact with the system by invoking one of the permanent application programs that have been written previously.

Examples:

Bank tellers check account balance daily.
Reservation clerks for airlines, hotels and car rental companies.

- **Sophisticated end users**: They interact with the system without writing programs. They form their requests in a DB query language and submitted to a query processor. Includes business analysts, scientists, and engineers, others thoroughly familiar with the system capabilities can implement their applications.

- **Standalone users**: Maintains personal databases by using ready-made/ready-to-use package applications that provide easy-to-use menu or graphics-based interfaces.
  Example: A tax program users that creates his or her own internal database.

Workers behind the Scene

- **DBMS system designers and implementers**
- **Tool developers**
- **Operators and maintenance personnel**

  - **DBMS system designers and implementers:** They are responsible for design and implement the DBMS modules and interfaces as a software package. The DBMS consists of many modules like implementing the catalog, processing query language, processing the interface, accessing and buffering data, controlling concurrency, and handling data recovery and security. The DBMS must interface with other system s/w(O.S and compilers) for various programming languages.

  - **Tool developers:** They are responsible for design and implement tools- the s/w packages that facilitate DB modeling and design, DB system design, improved performance monitoring, natural language or graphical interface, prototyping, simulation, and test data. Independent s/w vendors develop and market these tools.

  - **Operators and maintenance personnel**: The DBA are responsible for the actual running and maintenance of the hardware and software environment for the DB system.

Advantages of using Database Management System

- **Controlling Redundancy**: In traditional file processing, every user group maintains its own files for handling the data.

Example: Consider BANK database, which has two groups of users. The Customer saving file and checking account file. These files keep track of information of customer and account. The other group of people may duplicate some or all of the same data in their own files or at multiple places.

File1: (CS) CUSTOMER_SAVING

Account_number	Name	Age	Address
123467	Aruna	25	#45,Bangalore
123469	Suri	26	#35,hassan
123468	Mohan	30	#23,Bangalore

Storing same data fields

File2: (CA)CHECKING_ACCOUNT

Account_number	Name	Address	Amount
123467	Aruna	#45,Bangalore	24000
123469	Suri	#35,hassan	50000
123468	Mohan	#23,Bangalore	30000

Figure1.11 Data redundancy

Storing the same data multiple times leads to redundancy. Redundancy has several problems i.e.

1) Single logical update: When new customer is added, it must be saved twice in 'CS' and 'CA'. This leads to duplication of effort.

2) Storage space is wasted when the same data is stored repeatedly.

3) Files that represent the same data may become inconsistent. This may happen because an update is applied to only one file but not to other files as shown in below figure1.11. A user cannot judge which is correct in this two files the above said problems does not exist in the database. A database design stores each logical data item in only one place in the database. This ensures consistency and saves storage space in DB.

Updated name and address

File1: (CS) CUSTOMER_SAVING

Account_number	Name	Age	Address
123467	Aruna M G	25	#125,Bangalore
123469	Suri	26	#35,hassan
123468	Mohan	30	#23,Bangalore

Not updated name and address (Inconsistent state)

File2: (CA)CHECKING_ACCOUNT

Account_number	Name	Address	Amount
123467	Aruna	#45,Bangalore	24000
123469	Suri	#35,hassan	50000
123468	Mohan	#23,Bangalore	30000

Figure1.12 Inconsistent state

- **Sharing of data among multiple users:** Concurrently multiple users can share and access a large database.

- **Restricting Unauthorized Access:** When multiple users share a large database, each user requires different information. The users will not be authorized to access all information in the database. A DBMS provides a security and authorization subsystem, which DBA uses to create account and to specify account restriction. The type of access operation are retrieval, update and deleting data must be controlled by giving account numbers protected by passwords.

- **Providing persistent storage for program Objects:** DB can be used to provide persistent storage for program objects and data structures. Programming languages like Pascal, java or C++ the values of program variables are discarded/lost once a program terminates, till programmer explicitly stores them in permanent files.

- **Providing Storage Structures for efficient Query Processing**
  Database systems must provide capabilities for efficiently executing queries and updates. Because the database is stored on disk, the DBMS must provide specialized data structures to speed up disk search for the desired records. Auxiliary files called indexes are used for this purpose.

- **Providing backup and recovery services.**
  A DBMS must provide facilities for recovering from hardware or software failures. The backup & subsystem of the DBMS is responsible for recovery.
  Example: If system fails during the transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.

- **Providing multiple interfaces to different classes of users.**
  DBMS should provide a variety of user interfaces. They include query language for Casual users, programming language interface for application programmers, forms and command codes for parametric users, and menu-driven interfaces and natural language interfaces for standalone users.

- **Representing complex relationships among data.**
  A database may include numerous varieties of data that are interrelated in many ways. The DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationship as they arise & to retrieve & update related data easily & efficiently.

- **Enforcing integrity constraints on the database.**
  DBMS should provide capabilities for defining & enforcing these constraints. The simplest type of integrity constraint involves specifying a data type for each data item.
  Example: name – string of not more than 30 alphabets
  A more complex type of constraint is that a record in one file must be related to records in other file. For example, in figure 1.1

- **Permitting Inferences and Actions using rules**
  Some database system provide capabilities for defining deduction rules for inferencing new information from the stored data base facts such system are called deductive database systems.

- **Additional Implications of Using the database Approach**
  - Potential for Enforcing Standards
    The database permits the DBA to define and enforce standards among database users.
  - Reduced Application Development Time
    Once a database is up and running, substantially less time is generally required to create new applications using DBMS facilities.
  - Flexibility
    Database structure can be changed as per the requirements change without affecting the stored data and the existing application programs.
  - Availability of Up-to-Date Information
    As soon as one user's update is applied to the database, all other users can immediately see this update.
  - Economies of Scale
    Makes the organization profitable by combining many application & data.

A Brief History of Database ApplicationsData Models

Model

- A structure that demonstrates all the required features of the parts of the real world which is of interest to the users of the information in the model.
- Representation and reflection of the real world (Universe of Discourse)

Data Model

- A set of concepts that can be used to describe the structure of a database: the data types, relationships, constraints, semantics and operational behavior.
- It is a tool for data abstraction

A model is described by the schema which is held in the data dictionary.

Categories of Data Models

- **High-level or Conceptual data models**: Provides concepts that describes the structure of the database (based on entities, attributes and relationships)
  Entity: It represents a real world object such as an employee or a project.
  Attributes: It represents some property of interest that describes an entity, such as the employee's name or salary.
  Relationship: A relationship among the two or more entities represents an association among two or more entities.
  Example: a work-on relationship between an employee and a project
- **A low-level or physical data model:** Provides concept that describes the details of how data is stored in the computer. These concepts generally meant for computer specialists, not for typical end users.
- **Representational or implementation data models:** It is between the high level or low level data models, which provides concepts that is understood by end user and also the way data is organized by within the computer. It Hide some details of data storage but can be implemented on a computer system directly (record-based, object oriented).

The types of data models
a) Relational Model
b) Network Model
c) Hierarchical Model
d) Object Oriented Model
e) Semantic Data Model

a) Relational Model

The relational model uses a collection of tables to represent both data and the relationship among those data. Each table has multiple columns and each column has unique name.
Or
It represents a DB as collection of tables where each table can be stored as a separate file.

- Commercial systems include: ORACLE, DB2, SYBASE, INFORMIX, INGRES, SQL Server
- Dominates the database market on all platforms

Relational Model - Data Structures

- domains
- attributes
- relations

Relation name

flight-schedule

flight#:integer	airline:char(20)	weekday:char(2)	price:dec(6,2)

Attribute names domain names

Relational Model - Operations

- Powerful set-oriented query languages
- Relational Algebra: procedural; describes how to compute a query; operators like JOIN, SELECT, PROJECT
- Relational Calculus: declarative; describes the desired result, e.g. SQL, QBE
- Insert, delete, and update capabilities.

FLIGHT-SCHEDULE

FLIGHT#	AIRLINE	WEEKDAY	PRICE
101	delta	mo	156
545	american	we	110
912	scandinavian	fr	450
101	delta	thru	200

DEPT-AIRPORT

FLIGHT#	AIRPORT-CODE
101	atl
912	cph
545	lax

Figure1.13: Relational Model

EXAMPLE2: BANK DATABASE

ACCOUNTTABLE

Account_number	Balance
A101	500
123469	12000
123468	15000
A105	100
A123467	20000
A102	900

CUSTOMERTABLE

SSN	Name	CSTREET	CCITY	ACNO
500	Aruna	SBL	BANG	A101
263	Suri	LP	ASK	A123469
272	Mohan	LP	ASK	A123468
501	Jag	SBL	BANG	A105
258	Arun	RT	BANG	A123467
503	CHAL	LP	ASK	A102

ADVANTAGES:

- Data represent in a Simplified format.
- Manipulating record is simple with the use of key attributes.
- Representation of different types of relationship is possible with this model.

b) Network Model

Data in network model are represented by collections of records, and relationships, among data are represented by links. Which can viewed as pointers.

Or

The network model allows any unit to be related to other unit through fixed relationship.

- Based on the CODASYL-DBTG 1971 report
- Commercial systems include, CA-IDMS and DMS-1100

Example1; bank database

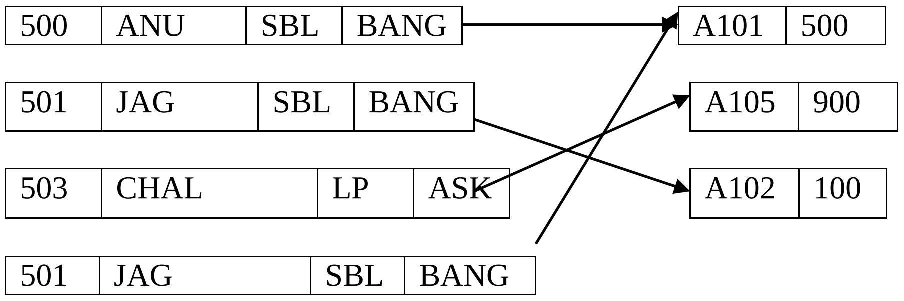
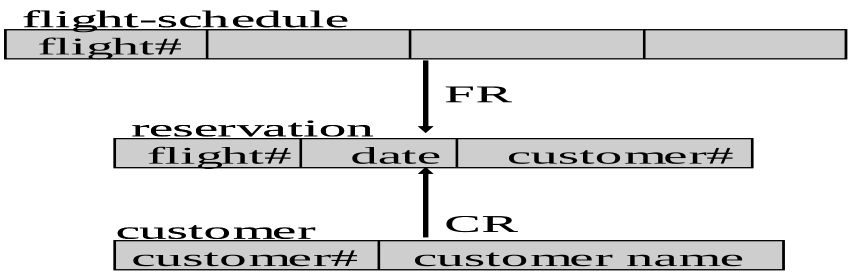


Figure1.14: Network Model

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

Example2; Airline Database

- owner record types: flight-schedule, customer
- member record type: reservations
- DBTG-set types: FR, CR
- n-m relationships cannot be modeled directly
- recursive relationships cannot be modeled directly



ADVANTAGES:
•Network Model is able to model complex relationships and represents semantics of add/delete on the relationships.
•Can handle most situations for modeling using record types and relationship types.
•Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET etc. Programmers can do optimal navigation through the database.

DISADVANTAGES:
•Navigational and procedural nature of processing
•Database contains a complex array of pointers that thread through a set of records.
Little scope for automated "query optimization”

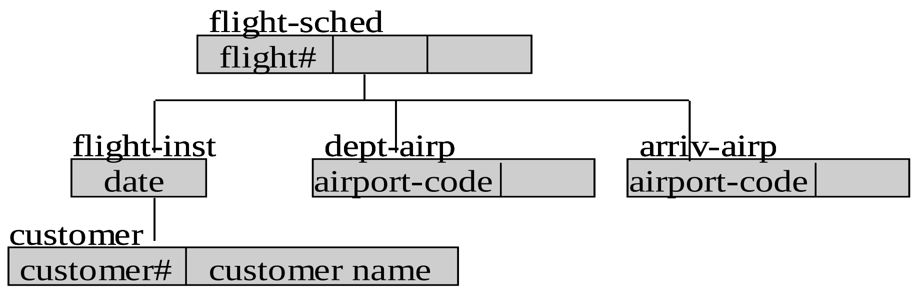
c) Hierarchical Model

The hierarchical model is similar to the network model in the sense that data and relationship among data are represented by record and link, respectively. It differs from the network model in that the records are organized as collections of trees rather than arbitrary graph.
Or
It used to create a relationship b/n two units. One record type is the root, all other record types is a child of one parent record type only.

- Commercial systems include IBM’s IMS, MRI’s System-2000 (now sold by SAS), and CDC’s MARS IV

Exapmle1;

- record types: flight-schedule, flight-instance, etc.
- field types: flight#, date, customer#, etc.
- parent-child relationship types (1:n only!!):
- (flight-sched,flight-inst), (flight-inst,customer)
- substantial duplication of customer instances
- asymmetrical model of n:m relationship types



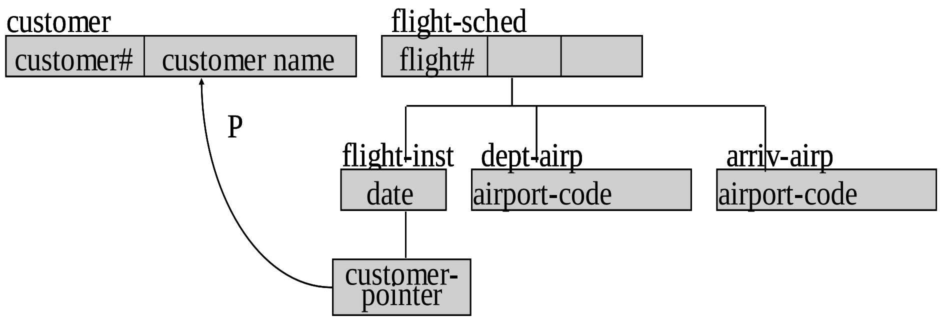
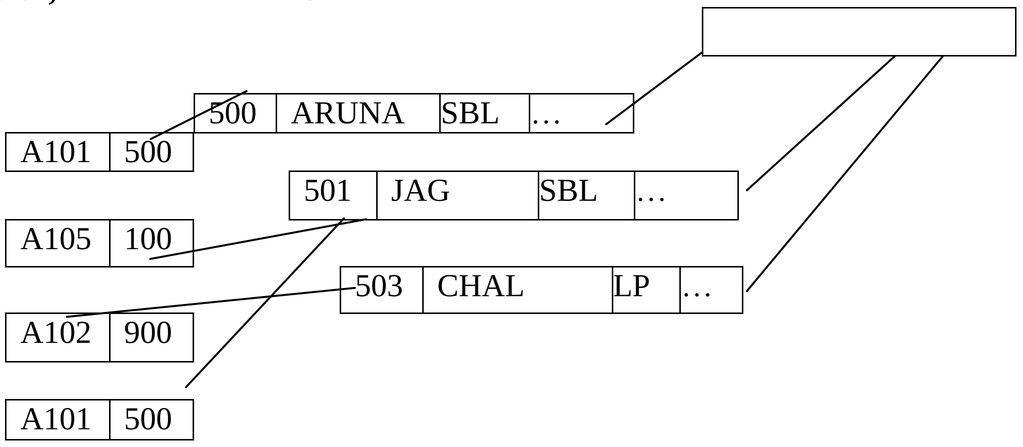


Figure1.15: Hierarchical Model

Example2; BANK DATABASE



Advantages:

- ❖ It is well suited for modeling hierarchical data.
- ❖ Top-down nature makes programming efficient.
- ❖ Hierarchical Model is simple to construct and operate on
- ❖ Corresponds to a number of natural hierarchically organized domains - e.g., assemblies in manufacturing, personnel organization in companies
- ❖ Language is simple; uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT etc

Disadvantages

- ❖ Biased to one way access from superiors to subordinates.
- ❖ Not suited for M:N R/n
- ❖ Difficult to insert subordinate records without superior records.

- ❖ Deleting superiors logically deletes subordinates.
- ❖ Difficult to update records stored in multiple location
- ❖ Not suited for DB whose R/n change.
- ❖ Navigational and procedural nature of processing
- ❖ Database is visualized as a linear arrangement of records
- ❖ Little scope for "query optimization

d) **The Object Oriented Model** (ODMG)

As a new family of higher-level implementation data models that are closer to conceptual data models.

Like the E-R model the object-oriented model is based on a collection of objects.
An object contains values stored in instance variables within the object. An object also contains bodies of codes that operate on the object. These bodies of code are called methods.

Objects that contain the same types of values and the same methods are grouped together into classes. A class may be viewed as a type definition for objects.

This combination of data and methods comprising a type definition is similar to a programming-language abstract data type.

The only way in which one object can access the data of another object is by invoking a method of that other object. This action is called sending a message to the object.

Example1: Consider an object representing a bank account. Such an object contains instance variables account-number and balance. It contains a method pay-interest which adds interest to the balance.

Or

- ➢ based on the object-oriented paradigm,
- ➢ e.g., Simula, Smalltalk, C++, Java
- ➢ area is in a state of flux
- ➢ object-oriented model has object-oriented repository model; adds persistence and database capabilities; (see ODMG-93, ODL, OQL)
- ➢ object-oriented commercial systems include GemStone, Ontos, Orion-2, Statice, Versant, O2
- ➢ object-relational model has relational repository model; adds object-oriented features; (see SQL3)
- ➢ object-relational commercial systems include Starburst, POSTGRES

Example2: OO syntax

```
class flight-schedule {
                        type tuple (flight#: integer,
                weekdays: set ( weekday: enumeration {mo,tu,we,th,fr,sa,su})
               dept-airport: airport, arriv-airport: airport)
                        method reschedule(new-dept: airport, new-arriv: airport)}
```

Schemas, Instances, and Database State

Database Schema: The description of a database is called database schema, which is specified during database design and is not expected to change frequently.

Student(studno,name,address)
Course(courseno,lecturer)

Schema

Student(123,Bloggs,Woolton)
Course(321,Jones,Owens)

Instance

Schema Diagram: A displayed schema is called a Schema diagram
Example:

FLIGHT-SCHEDULE

FLIGHT#	AIRLINE	WEEKDAY	PRICE

DEPT-AIRPORT

FLIGHT#	AIRPORT-CODE

Figure1.16: Schema diagram

Schema Construct: We call each object in the schema – such as FLIGHT-SCHEDULE or DEPT-AIRPORT

Database state: 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.

Empty state: The database with no data is called empty state.

Initial state: when the database is first populated or loaded with the initial data.

Current state: At any point in time, the database has a current state.

Valid state: A state that satisfies the structure and constraints specified in the schema.

Invalid state: If a state that do not satisfies the structure and constraints specified in the schema.

The schema is sometimes called the **intension**, and a database state is called an **extension** of the schema.

Three-Schema Architecture and Data Independence

The Three-Schema Architecture

- Architecture for DB systems called three-schema architecture.
- It used to help achieve and visualize these characteristics.
- The goal of the 3-schema architecture is to separate the user application and the physical DB.

1. Internal Level has and internal Schema, which describes the physical storage structure of the database. Uses a physical data model and describes the complete details of data storage and access path for the database.

2. Conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. Hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.

3. 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.

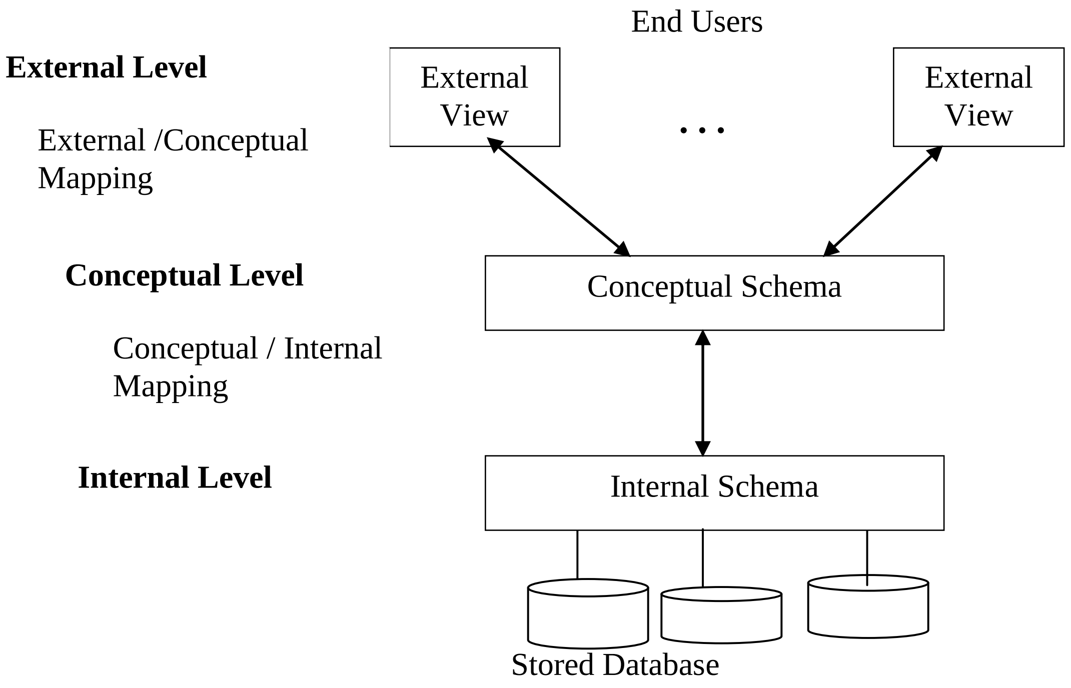


Figure1.17: The three-schema architecture

Data Independence

The capacity to change the schema at one level of a DB system without change the schema at the next higher level is called data independence. Two types of data independence:

1. **Logical data independence :** Change the conceptual schema without having to change the external schemas

2. **Physical data independence**: Change the internal schema without having to change the conceptual schema

Whenever we have a multiple-level DBMS, its catalog must be expanded to include information on how to map request and data among the various levels. Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed.

Database Languages and Interfaces

DBMS Languages:

Data definition language (DDL): It used by DBA and by DB designers to define both schemas. The DDL complier used to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog. It is used to specify the database conceptual schema only.

Storage definition language (SDL): It is used to specify the internal schema.
The mapping b/w the two schemas any be specified either one of these languages.

View definition language (VDL): It is used to specify user views and their mapping of data to storage. In relational DBMSs, SQL is used in the role of VDL to define user or application views as results of predefined queries.

Data manipulation language (DML): It used in manipulations of data like retrieval, insertion, deletion, and modification of the data.
There are two main types of DMLs. They are

1. **High-level or nonprocedural DML**
2. **Low-level or procedural DML**

High-level DML: Many DBMSs allow high-level DML statements either to be entered interactively from a display monitor or terminal or to be embedded in a general-purpose programming language.
Retrieve many records in a single DML statement; therefore, they are called **set-at-a-time or set-oriented DML's**.
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 or procedural DML: They must be embedded in a general-purpose programming language. The low-level are also called **record-at-a-time** because it retrieve only one record or objects at a time from the DB and processes each separately. It use programming language such as lopping, to retrieve and process each record from a set of records.
Ex: Hierarchical model is DL/1 uses command like GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT etc

Whenever DML commands, whether high level or low level, are embedded in general-purpose programming language, that language is called the **host language** and the DML is called **data sublanguage**. High-level DML used in a standalone interactive manner is called a **query language**.

DBMS Interfaces

Menu-Based Interfaces for Web Clients or Browsing. The query is composed step-by-step by picking options from a menu that is displayed by the system. Pull-down menus are very popular techniques in Web-based user interfaces. They are also often used in browsing interfaces, which allows a user to look through the contents of a database in an exploratory and unstructured manner.

Forms-Based Interfaces: Displays a form to each user. Users cal fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries. Forms are usually designed and programmed for naïve users as interfaces to canned transactions.

Graphical User Interfaces: A GUI typically displays a schema to the user in diagrammatic form. The user then can specify a query by manipulating the diagram. In many cases, GUIs utilize both menus and forms.

Natural Language Interfaces : These interfaces accept requests written in English or some other language and attempt to understand them. A naturally language interface usually has its own schema, which is similar to the database conceptual schema, as well as a dictionary of important words. The interface refers to the words in its schema, as well as to the set of standard words in its dictionary, to interpret the request. If the interpretation is successful, the interface generates a high-level query corresponding to the natural language request and submits it to the DBMS for processing; otherwise a dialogue is started with the user to clarify the request.

Speech Input and Output : The speech input is detected using a library of predefined words and used to set up the parameters that are supplied to the queries. For output, similar conversion form text or numbers into speech takes place.
Example: Applications with limited vocabularies such as inquiries for telephone directory, flight arrival/departure, and bank account information are allowing speech for input and output to enable ordinary folks to access this information.

Interfaces for parametric users: Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly.
Example: a teller is able to use single function keys to invoke routine and repetitive transactions such as deposits or withdrawals into accounts, or balance inquiries.
Usually a small set of abbreviated commands is included, with the goal of minimizing the number of keystrokes required for each request. This allows the parametric user to proceed with a minimal number of keystrokes.

Interfaces for the DBA : Most database systems contain privileged commands that can be used only by the DBS's staff. These include commands for creating accounts, setting parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

The Database System Environment:

DBMS Component Modules:

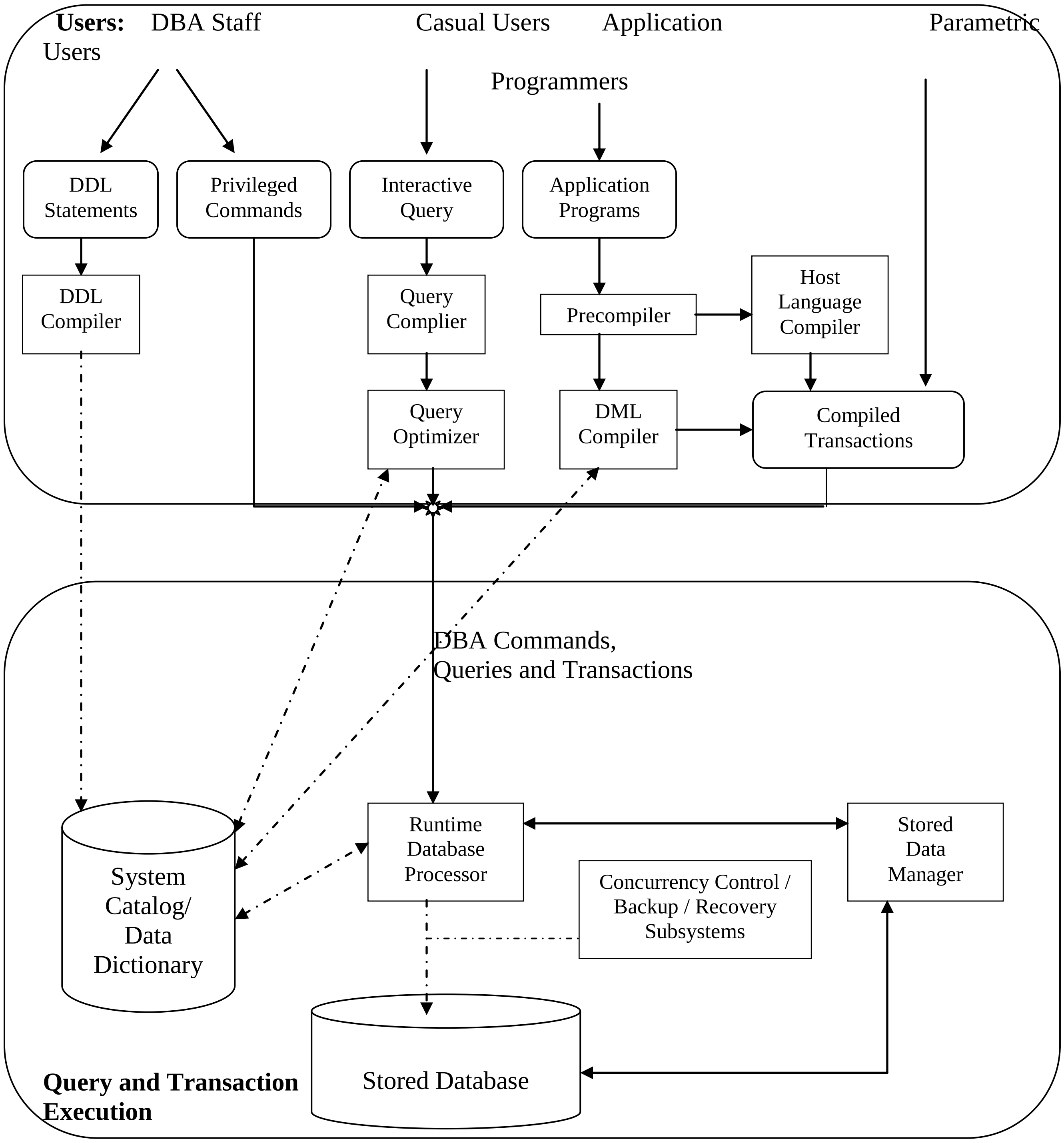


Figure1.18: Component modules of a DBMS and their interactions.

The **Top of the figure** refers to the various users of the database environment and their interfaces. Shows the interfaces for the DBA staff, casual users who work with interactive interfaces to formulate queries, application programmers who program using some host languages, and parametric users who do data entry work by supplying parameters to predefined transactions.

The **lower half shows** the internals of the DBMS responsible for storage of data and processing of transactions.

The **DBA staff works** on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands.
DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.

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 to many other types of information that are needed by the DBMS modules

Casual users and persons with occasional need for information from the database interact using some form of interface, **interactive query interface**. These queries are parsed, analyzed for correctness of the operations for the model, the name of data elements, and so on by a **query compiler** that compiles them into an internal form.

Query optimizer is concerned with 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.

Precompiler 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 programming is sent to the **host language compiler**

In lower half of the figure the runtime database processor is shown to execute
1. The privileged commands, 2. The executable query plans, and 3. The canned transactions with runtime parameters. It works with the system dictionary and may update it with statistics. It works with the stored data manager, for carrying out low-level input/output operations between disk and memory, management of buffers in main memory. Concurrency control and backup and recovery systems separately as module for transaction management.

Database System Utilities

Common utilities have the following types of functions:

- Loading – is used to load existing data files into the database, transferring data from one DBMS to another, loading programs are called conversion tools.
- Backup – dumping the entire database onto tape, this backup can be used to restore the database.

- Database storage reorganization – reorganize a set of database files into a different file organization to improve performance.
- Performance monitoring – monitors database usage and provides information to the DBA
- Other utilities for sorting files, handling data compression, monitoring access by users, interfacing with the network, and performing other functions.

Tools, Application Environments, and Communications Facilities

Tools – CASE tools used in design phase of database systems.
The other tool **Data dictionary / data repository** is used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc. Such system called **information repository**.
***Active* data dictionary** is accessed by DBMS software and users/DBA,
***Passive* data dictionary** is accessed by users/DBA only.

Centralized and client/Server Architectures for DBMSs

1) Centralized DBMSs Architecture

-Architecture for DBMS has followed trends similar to general computer system architecture.
-Earlier architectures used mainframe computers to provide the main processing for all system functions and DBMS functionality.
-The reason was that users used such system do not have processing power and only display terminals. So, all processing was done remotely on the computer system, and display information and controls were sent from the computer to display terminals which were connected to the central computer to communication network.
-As prices of H/w decreased, users replaced their terminals with PCs and workstations.
-At first DBMSs was used these concepts which was called Centralized DBMS. In this all the functionality, application programming execution and user interface processing were carried out on one machine. Below Figure 1.19 shows the components in a centralized architecture.

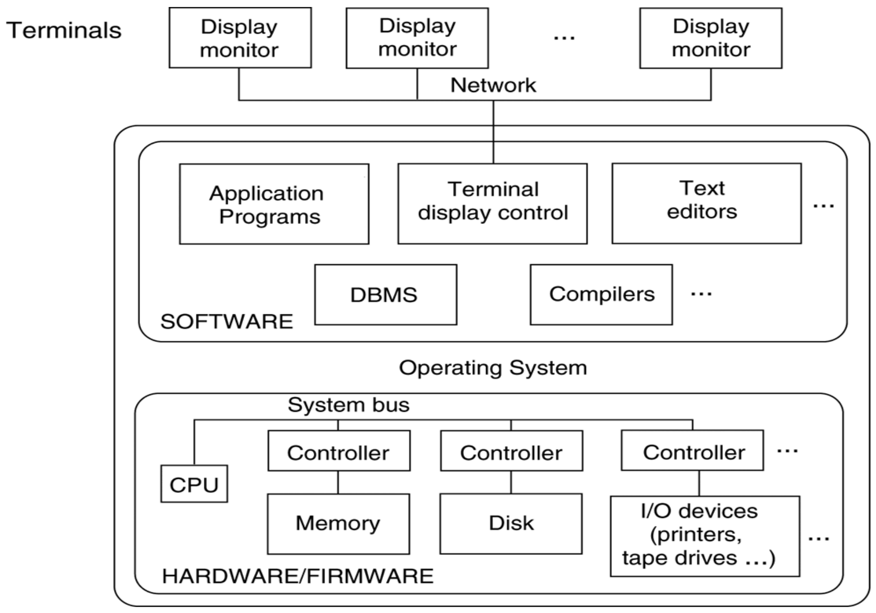


Figure 1.19: Centralized Architecture

2) Basic Client/Server Architectures

The client/server architecture was developed to deal with computing environment in which a large number of PC, workstation, printer server, and file server are connected through a network.

The idea is to define specialized servers with specific functionalities.

Example: Connect number of PC or small workstation as clients to file server that maintains files of the client machines. All print requests send by the clients are forward to printer server which is connected to various printers.

The client m/c provides the user with the appropriate interfaces to utilize these servers, as well as with local processing power to run local applications. Figure1.20 shows Client/server architecture at logical level with multiple clients and specific data stored on specific server and Figure1.21 shows Client/server architecture at physical level where some m/c are client sites, other m/c are server and still other m/c will have both client /servers functionality.

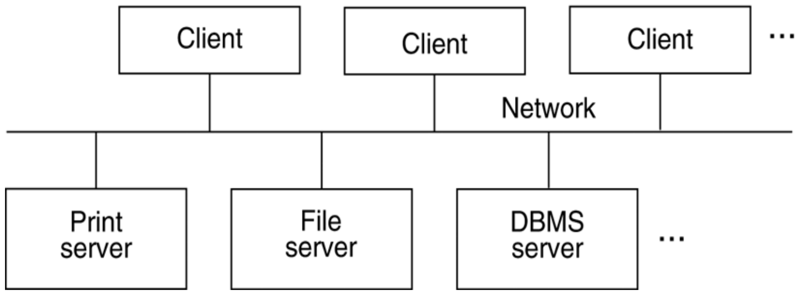


Figure1.20: Logical two-tier client/server architecture

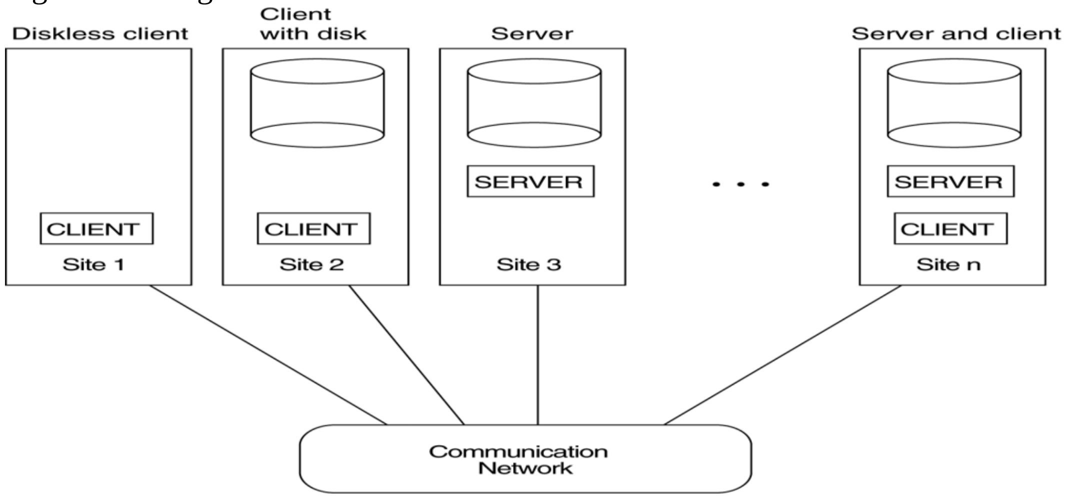


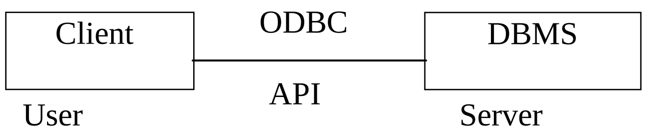
Figure1.21: Physical two-tier client/server architecture

A client in this framework is typically a user m/c (machine) that provides user interface capabilities and local processing. When client requires access to DB that does not exist at that m/c, it connects to server that provides the needed functionality.

A server is a system containing both H/w and S/w that can provide services to the client m/c.

Two-Tier Client/Server Architectures for DBMSs

-The client/server architecture is increasingly being incorporated into commercial DBMS packages.
-SQL provided a standard language for RDBMS, this created a logical dividing point b/w client and server. In RDBMS server is also called SQL server.
-The query and transaction functionality related to SQL processing remained/placed on the server.
-Server is also called a query server or transaction server.
-The user interface programs and application programs can run on the client side.
-When DBMS access is required , the program establishes connection to DBMS(server side); once connection is created, the client program can communicate with the DBMS.
-TO established connection b/w client and server the open database connectivity(ODBC) provides an application program interface(API).



-A client program can send query using ODBC API, which are then processed at the server site. Any query results are sent back to the client program.
-The architecture described here are called two-tier architecture because the s/w components are distributed over two systems client and server.
Advantages of Two-Tier:

1. Simplicity
2. seamless compatibility with existing systems

Three-Tier and N-tier Architecture for Web Application
Many web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the DB server.

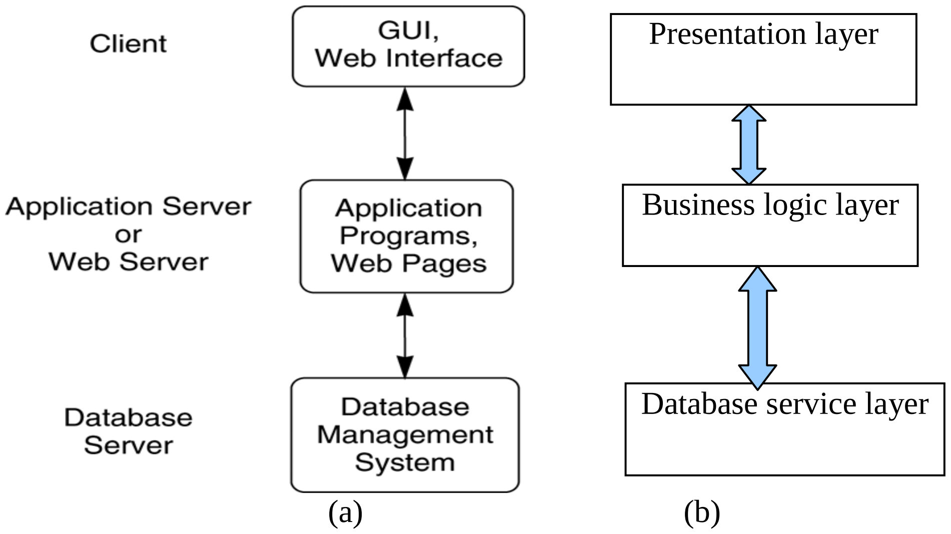


Figure1.22 Logical three-tier client/server architecture, with a couple of commonly used nomenclatures

The intermediate layer or middle tier called the application server and sometimes the web server depending on the application.
-The server plays an intermediary role by storing business rules(procedure or constraints) that are used to access data from DB server.
-It can improve security.
-Client contain GUI interfaces.
-The intermediate server accepts request from client, process the request and sends DB commands to the DB server, where it may processed further and output is passed to the users.
-Thus the user interface, application rule and data access act as the three-tiers.
-Figure1.22(b) shows another architecture used by DB and other application package vendors.
-The presentation layer display information to the user and allows data entry.
-The business logic layer handles intermediate rules and constraints before data is passed up top the user or down to the DBMS.
-The bottom layer includes all data management services.
-The layer b/w user and stored data can be derived in to finer components there by giving rise to n-tier architecture.
-Where n may be four or five.

Classification of DBMSs

1. **Based on the data model**:
- Traditional: Relational, Network, Hierarchical.
- Emerging: Object-oriented, Object-relational, extended Markup language model.

2. **Based on number of users:**
- **Single-user (typically used with micro- computers) vs. Multi-user system (most DBMSs):**
  Single user systems support only one user at a time.
  Multiuser user systems support multiple users at a time.

3**. Based on number of sites**:
- Centralized DBMS(uses a single computer with one database) vs. distributed DBMS(uses multiple computers, multiple databases).
- The types of DDBMS: Homogenous DDBMSs use the same DBMS s/w at multiple sites.
  Heterogeneous DDBMSs use the different DBMS s/w at multiple sites.

4**. DBMSs based on cost**:
- Some are open source (free) DBMS products available in n/w sites. The other type are sold in the form of licenses- site licenses allows unlimited use of the DB system with any number of copies running at the customer site. Another type of license limits the number of concurrent users at a location.

5. **Based on types of access path**: accessing/Storing files in the DBMS.
6. **DBMS can be general purpose or special purpose**:
- Ex: Airline reservations, Railway reservations and telephone directory systems are special purpose DBMSs. These fall under the category of online transaction processing (OLTP) systems.