Unit 1

1.1 Introduction

Data - Data is meaningful known raw facts that can be processed and stored as information.

Database - Database is a collection of interrelated and organized data. In general, it is a collection of files (tables).

DBMS - Database Management System (DBMS) is a collection of interrelated data [usually called database] and a set of programs to access, update and manage those data [which form part of management system]

OR

It is a software package to facilitate creation and maintenance of computerized database.

Importance: Database systems have become an essential component of life in modern society, in that many frequently occurring events trigger the accessing of at least one database: bibliographic library searches, bank transactions, hotel/airline reservations, grocery store purchases, online (Web) purchases, etc., etc.

Traditional vs. more recent applications of databases:

The applications mentioned above are all "traditional" ones for which the use of rigidly-structured textual and numeric data suffices. Recent advances have led to the application of database technology to a wider class of data. Examples include **multimedia** databases (involving pictures, video clips, and sound messages) and **geographic** databases (involving maps, satellite images).

Also, database search techniques are applied by some WWW search engines.

Definitions

The term **database** is often used, rather loosely, to refer to just about any collection of related data. E&N say that, in addition to being a collection of related data, a database must have the following properties:

- It represents some aspect of the real (or an imagined) world, called the **miniworld** or **universe of discourse**. Changes to the miniworld are reflected in the database. Imagine, for example, a UNIVERSITY miniworld concerned with students, courses, course sections, grades, and course prerequisites.
- It is a logically coherent collection of data, to which some meaning can be attached. (Logical coherency requires, in part, that the database not be self-contradictory.)
- It has a purpose: there is an intended group of users and some preconceived applications that the users are interested in employing.

To summarize: a database has some source (i.e., the miniworld) from which data are derived, some degree of interaction with events in the represented miniworld (at least insofar as the data is updated when the state of the miniworld changes), and an audience that is interested in using it.

An Aside: data vs. information vs. knowledge: Data is the representation of "facts" or "observations" whereas information refers to the meaning thereof (according to some interpretation). Knowledge, on the other hand, refers to the ability to use information to achieve intended ends.

Computerized vs. manual: Not surprisingly (this being a CS course), our concern will be with computerized database systems, as opposed to manual ones, such as the card catalog-based systems that were used in libraries in ancient times (i.e., before the year 2000). (Some authors wouldn't even recognize a non-computerized collection of data as a database, but E&N do.)

Size/Complexity: Databases run the range from being small/simple (e.g., one person's recipe database) to being huge/complex (e.g., Amazon's database that keeps track of all its products, customers, and suppliers).

Definition: A **database management system** (DBMS) is a collection of programs enabling users to create and maintain a database.

More specifically, a DBMS is a general purpose software system facilitating each of the following (with respect to a database):

- **definition:** specifying data types (and other constraints to which the data must conform) and data organization
- **construction:** the process of storing the data on some medium (e.g., magnetic disk) that is controlled by the DBMS
- **manipulation:** querying, updating, report generation
- **sharing**: allowing multiple users and programs to access the database "simultaneously"
- **system protection:** preventing database from becoming corrupted when hardware or software failures occur
- **security protection:** preventing unauthorized or malicious access to database.

Given all its responsibilities, it is not surprising that a typical DBMS is a complex piece of software.
A database together with the DBMS software is referred to as a **database system**.

An Example:

UNIVERSITY database
Among the main ideas illustrated in this example is that each file/relation/table has a set of named fields/attributes/columns, each of which is specified to be of some data type. (In addition to a data type, we might put further restrictions upon a field, e.g., GRADE_REPORT must have a value from the set {'A', 'B', ..., 'F'}.)
The idea is that, of course, each table will be populated with data in the form of records/tuples/rows, each of which represents some entity (in the miniworld) or some relationship between entities.
For example, each record in the **STUDENT** table represents a student. Similarly for the **COURSE** and **SECTION** tables.

Database manipulation involves querying and updating.

Examples of (informal) queries:

- Retrieve the transcript(s) of student(s) named 'Smith'.
- List the names of students who were enrolled in a section of the 'Database' course in Spring 2006, as well as their grades in that course section.
- List all prerequisites of the 'Database' course.

Examples of (informal) updates:

- Change the CLASS value of 'Smith' to sophomore (i.e., 2).
- Insert a record for a section of 'File Processing' for this semester.
- Remove from the prerequisites of course 'CMPS 340' the course 'CMPS 144'.

Of course, a query/update must be conveyed to the DBMS in a precise way (via the query language of the DBMS) in order to be processed.

As with software in general, developing a new database (or a new application for an existing database) proceeds in phases, including **requirements analysis** and various levels of **design** (conceptual (e.g., Entity-Relationship Modeling), logical (e.g., relational), and physical (file structures)).

Characteristics of the Database Approach:

Database approach vs. File Processing approach: Consider an organization/enterprise that is organized as a collection of departments/offices. Each department has certain data processing "needs", many of which are unique to it. In the **file processing approach**, each department would control a collection of relevant data files and software applications to manipulate that data.

For example, a university's Registrar's Office would maintain data (and programs) relevant to student grades and course enrollments. The Bursar's Office would maintain data (and programs) pertaining to fees owed by students for tuition, room and board, etc. (Most likely, the people in these offices would not be in direct possession of their data and programs, but rather the university's Information Technology Department would be responsible for providing services such as data storage, report generation, and programming.)

One result of this approach is, typically, **data redundancy**, which not only wastes storage space but also makes it more difficult to keep changing data items consistent with one another, as a change to one copy of a data item must be made to all of them (called **duplication-of-effort**). **Inconsistency** results when one (or more) copies of a datum are changed but not others. (E.g., If

you change your address, informing the Registrar's Office should suffice to ensure that your grades are sent to the right place, but does not guarantee that your next bill will be, as the copy of your address "owned" by the Bursar's Office might not have been changed.)

In the **database approach**, a single repository of data is maintained that is used by all the departments in the organization. (Note that "single repository" is used in the logical sense. In physical terms, the data may be distributed among various sites, and possibly mirrored.)

Main Characteristics of database approach:

1. **Self-Description:** A database system includes —in addition to the data stored that is of relevance to the organization— a complete definition/description of the database's structure and constraints. This **meta-data** (i.e., data about data) is stored in the so-called **system catalog**, which contains a description of the structure of each file, the type and storage format of each field, and the various constraints on the data (i.e., conditions that the data must satisfy).
   The system catalog is used not only by users (e.g., who need to know the names of tables and attributes, and sometimes data type information and other things), but also by the DBMS software, which certainly needs to "know" how the data is structured/organized in order to interpret it in a manner consistent with that structure. Recall that a DBMS is general purpose, as opposed to being a specific database application. Hence, the structure of the data cannot be "hard-coded" in its programs (such as is the case in typical file processing approaches), but rather must be treated as a "parameter" in some sense.

2. **Insulation between Programs and Data; Data Abstraction:**

   Program-Data Independence: In traditional file processing, the structure of the data files accessed by an application is "hard-coded" in its source code. (E.g., Consider a file descriptor in a COBOL program: it gives a detailed description of the layout of the records in a file by describing, for each field, how many bytes it occupies.)

   If, for some reason, we decide to change the structure of the data (e.g., by adding the first two digits to the YEAR field, in order to make the program Y2K compliant!), **every** application in which a description of that file's structure is hard-coded must be changed!

   In contrast, DBMS access programs, in most cases, do not require such changes, because the structure of the data is described (in the system catalog) separately from the programs that access it and those programs consult the catalog in order to ascertain the structure of the data (i.e., providing a means by which to determine boundaries between records and between fields within records) so that they interpret that data properly.

   In other words, the DBMS provides a conceptual or logical view of the data to application programs, so that the underlying implementation may be changed without the programs being modified. (This is referred to as program-data independence.)

   Also, which access paths (e.g., indexes) exist are listed in the catalog, helping the DBMS to determine the most efficient way to search for items in response to a query.

Data Abstraction:

- A **data model** is used to hide storage details and present the users with a conceptual view of the
- database.
  Programs refer to the data model constructs rather than data storage details

Note: In fairness to COBOL, it should be pointed out that it has a COPY feature that allows different application programs to make use of the same file descriptor stored in a "library". This provides some degree of program-data independence, but not nearly as much as a good DBMS does. End of note.

Example by which to illustrate this concept: Suppose that you are given the task of developing a program that displays the contents of a particular data file. Specifically, each record should be displayed as follows:

Record #i:
value of first field
value of second field
...
...
value of last field

To keep things very simple, suppose that the file in question has fixed-length records of 57 bytes with six fixed-length fields of lengths 12, 4, 17, 2, 15, and 7 bytes, respectively, all of which are ASCII strings. Developing such a program would not be difficult. However, the obvious solution would be tailored specifically for a file having the particular structure described here and would be of no use for a file with a different structure.

Now suppose that the problem is generalized to say that the program you are to develop must be able to display any file having fixed-length records with fixed-length fields that are ASCII strings. Impossible, you say? Well, yes, unless the program has the ability to access a description of the file's structure (i.e., lengths of its records and the fields therein), in which case the problem is not hard at all. This illustrates the power of metadata, i.e., data describing other data.

3. **Multiple Views of Data:** Different users (e.g., in different departments of an organization) have different "views" or perspectives on the database. For example, from the point of view of a Bursar's Office employee, student data does not include anything about which courses were taken or which grades were earned. (This is an example of a **subset** view.)

As another example, a Registrar's Office employee might think that GPA is a field of data in each student's record. In reality, the underlying database might calculate that value each time it is needed. This is called **virtual** (or **derived**) data.

A view designed for an academic advisor might give the appearance that the data is structured to point out the prerequisites of each course.

A good DBMS has facilities for defining multiple views. This is not only convenient for users, but also addresses security issues of data access. (E.g., The Registrar's Office view should not provide any means to access financial data.)

4. **Data Sharing and Multi-user Transaction Processing:** As you learned about (or will) in the OS course, the simultaneous access of computer resources by multiple users/processes is a major source of complexity. The same is true for multi-user DBMS's.

   Arising from this is the need for **concurrency control**, which is supposed to ensure that several users trying to update the same data do so in a "controlled" manner so that the results of the updates are as though they were done in some sequential order (rather than interleaved, which could result in data being incorrect).

   This gives rise to the concept of a **transaction**, which is a process that makes one or more accesses to a database and which must have the appearance of executing in isolation from all other transactions (even ones that access the same data at the "same time") and of being atomic (in the sense that, if the system crashes in the middle of its execution, the database contents must be as though it did not execute at all).

   Applications such as **airline reservation systems** are known as **online transaction processing** applications.

: **Actors on the Scene**

These apply to "large" databases, not "personal" databases that are defined, constructed, and used by a single person via, say, Microsoft Access.

Users may be divided into

Those who actually use and control the database content, and those who design, develop and maintain database applications (called ―Actors on the Scene‖), and
Those who design and develop the DBMS software and related tools, and the computer systems operators (called ―Workers Behind the Scene‖).

1. **Database Administrator** (DBA): This is the chief administrator, who oversees and manages the database system (including the data and software). Duties include authorizing users to access the database, coordinating/monitoring its use, acquiring hardware/software for upgrades, etc. In large organizations, the DBA might have a support staff.
2. **Database Designers**: They are responsible for identifying the data to be stored and for choosing an appropriate way to organize it. They also define **views** for different categories of users. The final design must be able to support the requirements of all the user sub-groups.

3. **End Users**: These are persons who access the database for **querying**, **updating**, and **report generation**. They are main reason for database's existence!
    - **Casual end users:** use database occasionally, needing different information each time; use query language to specify their requests; typically middle- or high-level managers.
    - **Naive/Parametric end users**: Typically the biggest group of users; frequently query/update the database using standard **canned transactions** that have been carefully programmed and tested in advance. Examples:
        - bank tellers check account balances, post withdrawals/deposits
        - reservation clerks for airlines, hotels, etc., check availability of seats/rooms and make reservations.
        - shipping clerks (e.g. ,at UPS) who use buttons, bar code scanners, etc., to update status of in-transit packages.
    - **Sophisticated end users**: engineers, scientists, business analysts who implement their own applications to meet their complex needs.
    - **Stand-alone users**: Use "personal" databases, possibly employing a special-purpose (e.g., financial) software package. Mostly maintain personal databases using ready-to-use packaged applications.
    - An example is a tax program user that creates its own internal database.
    - Another example is maintaining an address book
4. **System Analysts, Application Programmers, Software Engineers**:
    - **System Analysts**: determine needs of end users, especially naive and parametric users, and develop specifications for canned transactions that meet these needs.
    - **Application Programmers**: Implement, test, document, and maintain programs that satisfy the specifications mentioned above.

Workers Behind the Scene

- **DBMS system designers/implementors:** provide the DBMS software that is at the foundation of all this!
- **Tool developers**: design and implement software tools facilitating database system design, performance monitoring, creation of graphical user interfaces, prototyping, ets.
- **Operators and maintenance personnel**: responsible for the day-to-day operation of the system.

Capabilities/Advantages of DBMS's

1. **Controlling Redundancy:** Data redundancy (such as tends to occur in the "file processing" approach) leads to **wasted storage space**, **duplication of effort** (when multiple copies of a datum need to be updated), and a higher likelihood of the introduction of **inconsistency**.

   On the other hand, redundancy can be used to improve performance of queries. Indexes, for example, are entirely redundant, but help the DBMS in processing queries more quickly.

2. **Restricting Unauthorized Access:** A DBMS should provide a **security and authorization subsystem**, which is used for specifying restrictions on user accounts. Common kinds of restrictions are to allow read-only access (no updating), or access only

to a subset of the data (e.g., recall the Bursar's and Registrar's office examples from above).

3. **Providing Persistent Storage for Program Objects:** Object-oriented database systems make it easier for complex runtime objects (e.g., lists, trees) to be saved in secondary storage so as to survive beyond program termination and to be retrievable at a later time.
4. **Providing Storage Structures for Efficient Query Processing:** The DBMS maintains indexes (typically in the form of trees and/or hash tables) that are utilized to improve the execution time of queries and updates. (The choice of which indexes to create and maintain is part of physical database design and tuning and is the responsibility of the DBA.

The **query processing and optimization** module is responsible for choosing an efficient query execution plan for each query submitted to the system.

5. **Providing Backup and Recovery:** The subsystem having this responsibility ensures that recovery is possible in the case of a system crash during execution of one or more trans**ac**tions.

Providing Multiple User Interfaces: For example, query languages for casual users, programming language interfaces for application programmers, forms and/or command codes for parametric users, menu-driven interfaces for stand-alone users.

Representing Complex Relationships Among Data: A DBMS should have the capability to represent such relationships and to retrieve related data quickly.

Enforcing Integrity Constraints: Most database applications are such that the semantics (i.e., meaning) of the data require that it satisfy certain restrictions in order to make sense. Perhaps the most fundamental constraint on a data item is its data type, which specifies the universe of values from which its value may be drawn. (E.g., a Grade field could be defined to be of type Grade_Type, which, say, we have defined as including precisely the values in the set { "A", "A-", "B+", ..., "F" }.

Another kind of constraint is referential integrity, which says that if the database includes an entity that refers to another one, the latter entity must exist in the database. For example, if (R56547, CIL102) is a tuple in the Enrolled_In relation, indicating that a student with ID R56547 is taking a course with ID CIL102, there must be a tuple in the Student relation corresponding to a student with that ID.

Permitting Inferencing and Actions Via Rules: In a **deductive** database system, one may specify declarative rules that allow the database to infer new data! E.g., Figure out which students are on academic probation. Such capabilities would take the place of application programs that would be used to ascertain such information otherwise.

Active database systems go one step further by allowing "active rules" that can be used to initiate actions automatically.

A Brief History of Database Applications

Early Database Applications:

The Hierarchical and Network Models were introduced in mid 1960s and dominated during the seventies.
A bulk of the worldwide database processing still occurs using these models.
Relational Model based Systems:
Relational model was originally introduced in 1970, was heavily researched and experimented with in IBM Research and several universities.
Object-oriented and emerging applications:
Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.

Their use has not taken off much.

Many relational DBMSs have incorporated object database concepts, leading to a new category called object-relational DBMSs (ORDBMSs)
Extended relational systems add further capabilities (e.g. for multimedia data, XML, and other data types)

Relational DBMS Products emerged in the 1980s
Data on the Web and E-commerce Applications

- Web contains data in HTML (Hypertext markup language) with links among pages.
- This has given rise to a new set of applications and E-commerce is using new standards like XML (eXtended Markup Language).
- Script programming languages such as PHP and JavaScript allow generation of dynamic Web pages that are partially generated from a database
  - New functionality is being added to DBMSs in the following areas:
  - Scientific Applications
  - XML (eXtensible Markup Language)
  - Image Storage and Management
  - Audio and Video data management
  - Data Warehousing and Data Mining
- Spatial data management
  - Time Series and Historical Data Management
- The above gives rise to new research and development in incorporating new data types, complex data structures, new operations and storage and indexing schemes in database systems.
  - Also allow database updates through Web pages

1. **Defining:** Specifying data types and structures, and constraints for data to be stored.
2. **Constructing**: Storing data in a storage medium.
3. **Manipulating:** Involves querying, updating and generating reports.
4. **Sharing**: Allowing multiple users and programs to access data simultaneously.

Eg. Of DBMS- Access, dBase, FileMaker Pro, and FoxBASE, ORACLE etc.

Primary goals of DBMS are:

1. To provide a way to store and retrieve database information that is both convenient and efficient.
2. To manage large and small bodies of information. It involves defining structures for storage of information and providing mechanism for manipulation of information.

3. It should ensure safety of information stored, despite system crashes or attempts at unauthorized access.
4. Ifdataaretobesharedamongseveralusers,thensystemshouldavoidpossible anomalous results.

Various views of Data

Data abstraction:

It can be summed up as follows.
1. When the DBMS hides certain details of how data is stored and maintained, it provides what is called as the abstract view of data.
2. This is to simplify user-interaction with the system.
3. Complexity (of data and data structure) is hidden from users through several levels of abstraction.

Data abstraction is used for following purposes:
1. To provide abstract view of data.
2. To hide complexity from user.
3. To simplify user interaction with DBMS.
4.

Levels of data abstraction:
There are three levels of data abstraction.
1. Physical level: It describes how a record (e. ., customer) is stored.
Features:
a) Lowest level of abstraction.
b) It describes how data are actually stored.
c) It describ s low-l v l omplex data structures in detail.
d) At this l v l, ffici nt algorithms to access data are defined.
2. Logical level: It d scrib s what data stored in database, and the relationships among

Features:
a) It is next-higher level of abstraction. Here whole Database is divided into small simple structures.
b) Users at this level need not be aware of the physical-level complexity used to implement the simple structures.
c) Here the aim is ease of use.
d) Generally, database administrators (DBAs) work at logical level of abstraction.

3. View level: Application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.
Features:
a) It is the highest level of abstraction.
b) It describes only a part of the whole Database for particular group of users.
c) This view hides all complexity.
d) It exists only to simplify user interaction with system.

e) The system may provide many views for the whole system.

Data Models

A data model is a collection of concepts that can be used to describe the structure of a database and provides the necessary means to achieve this abstraction whereas structure of a database means the data types, relationships and constraints that should hold on the data.

Collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints. The various data models that have been proposed fall into three different groups. Object based logical models, record-based logical models and physical models.

Object-Based Logical Models: They are used in describing data at the logical and view levels. They are characterized by the fact that they provide fairly flexible structuring capabilities and allow data constraints to be specified explicitly. There are many different models and more are likely to come. Several of the more wiely known ones are:

·The E-R model

·The object-oriented model

·The semantic data model

·The functional data model

The E-R Model

The (E-R) data model is based on a p rc ption of a real worker that consists of a collection of basic objects, called entiti s, and of r lationships among these objects.

The overall logical structure of a database can be expressed graphically by an E-R diagram. Which is built up by t following components:

·Rectangles, which represent entity sets

·Ellipses, which represent attributes

- Diamonds, which rprsnt relationships among entity sets
- Lines, which link attributs to entity sets and entity sets to relationships.

  E.g. suppose we have two entities like customer and account, then these two entities canbemodeled.

The Object-Oriented Model

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 code that operate on the object. These bodies of code are called methods.

Classes: It is the collection of objects which consist of the same types of values and the same methods.

E.g. account number & balance are instance variables; pay-interest is a method that uses the above two variables and adds interest to the balance.

Semantic Models

These include the extended relational, the semantic network and the functional models. They are characterized by their provision of richer facilities for capturing the

meaning of data objects and hence of maintaining database integrity. Systems based on these models exist in monotype for at the time of writing and will begin to filter through the next decade.

Record-Based Logical Models

Record based logical models are also used in describing data at the logical and view levels. In contrast to object-based data models, they are used both to specify the overall logical structures of the database, and to provide a higher-level escription of the implementation.

Record-based models are so named because the database is structured in fixed-format records of several types. Each record type defines a fixed number of fields, or attributes, and each field is usually of a fixed length.

The three most widely accepted record-bas d data models are the relational, network, and hierarchical models.

Relational Model

Therelationalmodelusesacollectionoftablestorepresentbothdataandtherelationshipsamong those data. Each table has multiple columns, and each column has a unique name as follows:

CUSTOMER (TABLE NAME)

Customer–name	Customer-street	Customer-City	Account-number
Johnsons	Alma	Pala Alto	A-101
Smith	North	Ryc	A-215
Hayes	Main	Harrison	A-102
Turner	Dutnam	Stanford	A-305
Johnson	Alma	PalaAlto	A-201
Jones	Main	Harrison	A-217
Lindsay	Park	Pittifield	A-222
Smith	North	Rye	A-201

SAMPLE RELATIONAL DATABASE

Network Model

Data in the network model is represented by collection of records, and relationship among data is represented by links, which can be viewed as pointers. The records in the database are organized as collections of arbitrary graphs.

Disadvantages of Network Data Model:

a. System complexity – In a network model, data are accessed one record at a time. This makes it essential for the database designers, administrators, and programmers to be familiar with the internal data structures to gain access to the data. Therefore, a user-friendly database management system cannot be created using the network model.

b. Lack of structural independence – Making structural modifications to the database is very difficult in the network database model as the data access method is navigational. Any changes made to the database structure require the application programs to be

modified before they can access data. Though the network atabase model achieves data independence, it still fails to achieve structural independence.

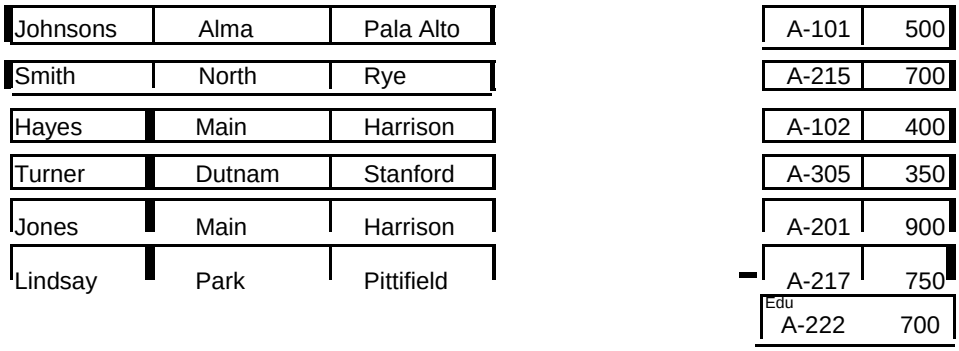

FIGURE 1.4: A SAMPLE NETWORK DATABASE

Hierarchical Model

The hierarchical model is similar to the network model in the sense that data and relationships among data are represented by records and links, respectively. It differs from the network model in that records are or anized as collection of trees rather than arbitrary graphs.

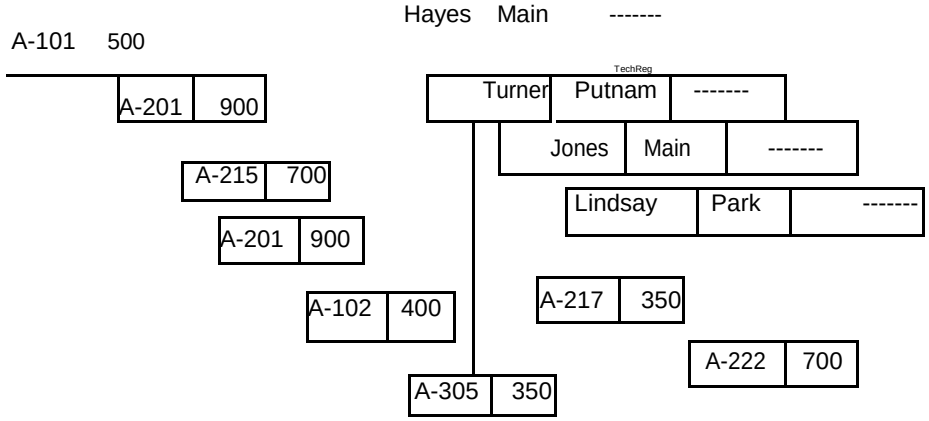

FIGURE : A SAMPLE HIERACHICAL ATABASE

Advantages of hierarchical Model:

Simplicity – Since the database is based on the hierarchical structure, the relationship between the various layers is logically simple. Thus, the design of a hierarchical database is simple.

DataSecurity–

Hierarchical model was the first database that offered the data security that is provided and enforced by the DBMS.

Data Integrity – Since the hierarchical model is based on the parent/child relationship, there is always a link between the parent segment and the child segment under it. The child segments are always automatically referenced to its parent, this model promotes data integrity.

Efficiency – The hierarchical database model is a very efficient one when the database contains a large number of one-to-many relationships and when the users require large number of transactions, using data whose relationships are fixed.

Physical Data Models

Physical data models are used to describe data at the lowest level. In contrast to logical data models, there are few physical data models in use. Two of the widely known ones are the unifying model and the frame-memory model.

1. Database systems are partitioned into modules for different functions. Some functions (e.g. file systems) may be provided by the operating system.
2. Broadly the functional components of a database system are:

a. Query Processor: It is one of the functional components of DBMS. It translates statements in a query language into low-level instructions the database manager understands. It may also attempt to find an equivalent but more efficient form.

It contains following components:

a. DML compiler - It converts DDL statements to a set of tables containing metadata stored in a data dictionary.

It also performs query optimization.

b. DDL interpreter – It interprets DDL statements and records definitions into data dictionary.

c. Query evaluation engine – It executes low-levelinstructionsgeneratedby DML compiler. They mainly deal with solving all problems related to queries and query processing. It helps database system simplify and facilitate access to data.

b. Storage Manger (Database Manager)

1. Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
2. The storage manager is responsible to the following tasks: 1. interaction with the file manager 2. efficient storing, retrieving and updating of data
3. The important components include:

a. File manager: It manages allocation of disk space and data structures used to represent information on disk.

b. Database manager: It is the interface between low-level data and remainsinaconsistent(corrct)state despite system failures (e.g., power failures and operating syst m crash s) and transaction failures.

d. DML precompiler: It conv rts DML statements embedded in an application program to normal procedure calls in a host language. The precompiler interacts with the query processor.

e. DDL ompiler: It converts DDL statements to a set of tables containing metadata stored in a data dictionary.

f. Authorization and integrity manger – It conducts integrity checks and user authority to access data.

g. Buff r manger – It is critical part of DB and stores temporary data.

In addition, several data structures are required for physical system implementation:

a. Data files: They store the database itself.

b. Datadictionary: Itstoresinformationaboutthestructureofthe

database. It is used heavily. Great emphasis should be placed on developing a good design and efficient implementation of the dictionary. In short, it stores metadata.

c. Indices: They provide fast access to data items holding particular values.

File systems/File processing systems

A file system is basically storing information in data structures called _files' in the operating system and manipulating this information via application programs that manipulate the files.

File Processing System

Advantages	**Disadvantages**
Simpler to use	Typicallydoesnotsupportmulti-user access
Less expensive· Fitstheneedsofmanysmallbusinesses and home users	Limited to smaller databases Limitedfunctionality(i.e.nosupportfor complicated transactions, recovery, etc.)
PopularFMS'sarepackagedalongwith theoperatingsystemsofpersonal computers(i.e.MicrosoftCardfileand Microsoft Works)	Decentralization ofata
Good for database solutions for hand held devices such as Palm Pilot TechReg	Redundancy and Integrity issues

Disadvantages of File Processing Syst m:

1. Data Redundancy – Since different programmers create the files and application programs over a long period, t various files are likely to have different formats and the programs may be written in several programming languages. Moreover, the same information may be dupli ated in several files, this duplication of data over several files is known as data redundan y. Eg. The address and telephone number of a particular customer may app ar in a file that consists of saving- account records and in a file that consists of checking account records. This redundancy leads to higher storage & excess cost also leads to inconsistency discussed in the next.

2. Data Inconsistency – The various copies of same data may no longer agree i.e. various copies of the same data may contain different information. Eg. A changed customer address may be reflected in savings-account records but not elsewhere in the system.

3. Difficulty in accessing data – In a conventional file processing system it is difficult to access the data in a specific manner and it is require creating an application program to carry out each new task. Eg. Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area. We ask the officer of data-processing department to generate such a list.We have 2 choices:

- List all cust_names & manually do the information required
- Ask the data processing dept. to have system programmer to write necessity application program Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it.

4. DataIsolation–Becausedataarescatteredinvariousfiles,andfilesmaybein different formats, writing new application programs to retrieve the appropriate data is difficult.

5. Integrity problems – The data stored in the database must satisfy certain types of consistency constraints. Eg. The balance of a bank account may never fall below a prescribed amount (say, ICICI 2500/-). Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

6. Atomicity problems – A computer system, like anyothermechanical or electrical device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.

and faster response,TechRegmanysystemsallowmultiple users to update the data simultaneously. In such an environment, int raction of concurrent updates may result in inconsistent data. Eg. Consider bank account A, containing $500. If two customers withdraw funds (say $50 and $100 respectively) from account A at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state. Suppose t at t programs executing on behalf of each withdrawal read the old balance, redu e t at value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value $500, and write back $450 and $400, r spectively. Depending on which one writes the value last, the account may contain ith r $450 or $400, rather than the correct value of $350.

To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

8. Security Problems – Not every user of the database system should be able to access all the data. Eg. In a bank system, payroll personnel need to see only that part of the database that has information about the various bank employees. They do not need access to information about customer accounts. But, since application programs are added to the system in an ad hoc manner, enforcing such security constraints is difficult.

Difference between DBMS and File-processing system:

DBMS	File-processing Systems
1. Redundancies and inconsistencies in data are reduced due to single file formats and duplication of data is eliminated.	1. Redundancies and inconsistencies in data existduetosinglefileformatsand duplication of data.
2. Data is easily accessed due to standard query procedures.	2.Datacannotbeeasilyaccesseddueto specialapplicationprogramsneededto access data.
3.Isolation/retrievalofrequireddatais possibleduetocommonfileformat,and there are provisions to easily retrieve data.	3. Data isolation is difficult due to different fileformats,andalsobecausenew application programs have to be written.
4. Integrity constraints, whether new or old, can be created or modified as per need. TechReg	4.Introductionofintegrityconstraintsis tediousandagainnewapplication prorams have to be written.
5. Atomicity of updates is possible.	5.Atomicityofupdatesmaynotbe maintained.
6. Several users can access data at tsame time i.e concurrently without problems	6. Concurrent accesses may cause problems such as . Inconsistencies.
7.Securityfeatursanbeenabledin DBMS very easily.	7.Itmaybedifficulttoenforcesecurity features.

Advantages of a DBMS over file-processing system:

A DBMS has three main features:

(1) Centralized data management,
(2) Data independence and
(3) Data integration /System integration

In a DB system, the DBMS provides the interface b/w the application programs & data. When changes are made to the data representation, the metadata maintained by

the DBMS is changed but the DBMS continues to provide data to application programs in previously used way. The DBMS handles the task of information of data where ever necessary. This independence b/w the programs & the data is called data independence. this made program to continue irrespective of changes made in it. To provide a high
degree of data independence, a DBMS must include a sophisticated metadata mgmt system. In DBMS , all files are integrated into one system thus reducing redundancies & making data mgmt more efficient. In addition, DBMS provides centralized control of the operational data. Some of advantages of above mention three features are:
Due to its centralized nature, the database system can overcome the disadvantages of the file-based system as discussed below.

• **Minimal Data Redundancy** - Since the whole data resides in one central database, the various programs in the application can access data in different data files. Hence data present in one file need not be duplicated in another.
This reduces data redundancy. However, this does not mean all redundancy can be eliminated. There could be business or technical reasons for having some amount of redundancy. Any such redundancy should be carefully controlled and the DBMS should be aware of it.
• **Data Consistency** - Reduced data redundancy leads tobetter data consistency.
centralized manner. TechReg Even
new applications can be developed to op rate against the same data.

• **Enforcement of Standards** - Enforcing standards in the organization and structure of data files is required and also easy in a Database System, since it is one single set of programs which is always intera ting with the data files.

• **Application Dev lopm nt Ease** - The application programmer need not build the functions for handling issu s like concurrent access, security, data integrity, etc. The programmer only n ds to implement the application business rules. This brings in application development ease. Adding additional functional modules is also easier than in file based systems.

• **Better Controls** - Better controls can be achieved due to the centralized nature of the system.

• **Data Independence** - The architecture of the DBMS can be viewed as a 3-level system comprising the following:

- The internal or the physical level where the data resides.
- The conceptual level which is the level of the DBMS functions
- The external level which is the level of the application programs or the end user.

Data Independence is isolating an upper level from the changes in the organization or

structure of a lower level. For example, if changes in the file organization of a data file do not demand for changes in the functions in the DBMS or in the application programs, data independence is achieved. Thus Data Independence can be defined as immunity of applications to change in physical representation and access technique. The provision of data independence is a major objective for database systems.

• **Reduced Maintenance** - Maintenance is less and easy, again, due to the centralized nature of the system.

Responsibility of Database Administrator

1.The database administrator is a person having central control over data and programs accessing that data. He coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.

2. Storage structure and access m thod d finition: writing a set of definitions translated by the data storage and definition language compiler.

3. Schema and physical organization modification: writing a set of definitions used by the DDL compiler to generate modifications to appropriate internal system tables (e.g. data dictionary). This is done rarely, but sometimes the database schema or physical organization must be modifi d.

4. Granting user authority to access the database: granting different types of authorization for data access to various users

5. Specifying integrity constraints: generating integrity constraints. These are consulted by the database manager module whenever updates occur.

6. Routine Maintenance: It includes the
following: a. Acting as liaison with users.
b. Monitoring performance and responding to changes in requirements.
c. Periodically backing up the database.

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.

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

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

- 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