Department of Commerce (CA)

CORE PAPER-II-DATABASE SYSTEM CONCEPTS

SEMESTER:I **SUB CODE:18MCC12C**

M.COM(CA)

UNIT1: Database system architecture-basic concepts-data system-operational data, data independence architecture for a database system-distributed databases.

REFERENCE BOOK:

An introduction to database system-C.J. Dates

An introduction to database system-Bipin

PREPARED BY: DR. E.N. KANJANA,

ASST PROFESSOR.

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases can be very large.
- Databases touch all aspects of our lives

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Levels of Abstraction

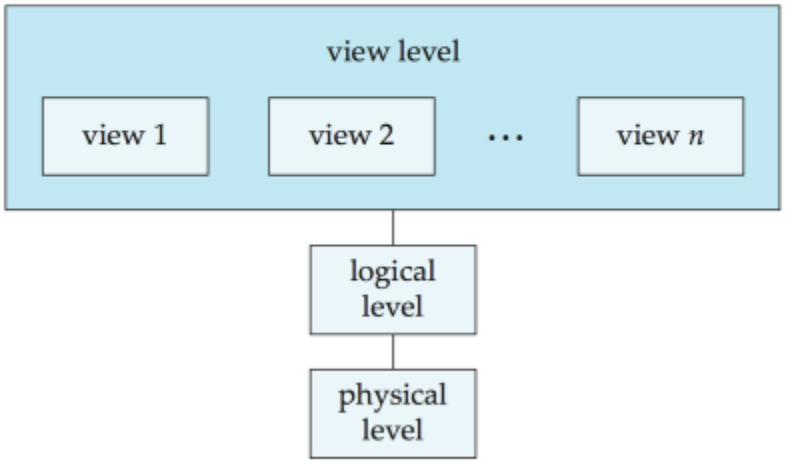
- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

```
type instructor = record
        ID : string;
        name : string;
        dept_name : string;
        salary : integer;
    end;
```

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

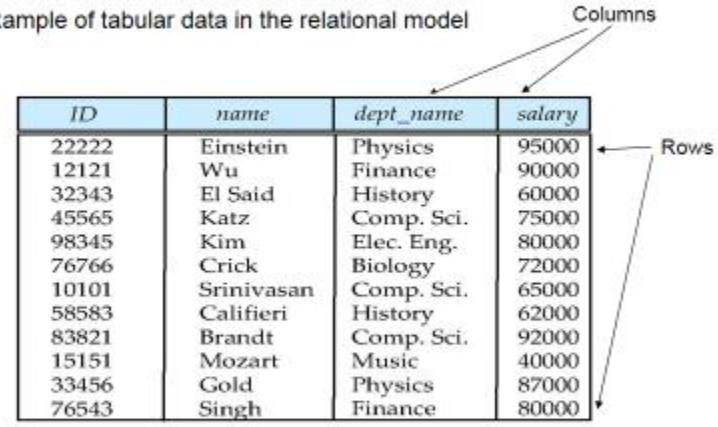
- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema**– the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Data Processing Vs. Data Management Systems

Although Data Processing and Data Management Systems both refer to functions that take raw data and transform it into usable information, the usage of the terms is very different. **Data Processing** is the term generally used to describe what was done by large mainframe computers from the late 1940's until the early 1980's (and which continues to be done in most large organizations to a greater or lesser extent even today): large volumes of raw transaction data fed into programs that update a master file, with fixedformat reports written to paper

The term **Data Management Systems** refers to an expansion of this concept, where the raw data, previously copied manually from paper to punched cards, and later into dataentry terminals, is now fed into the system from a variety of sources, including ATMs, EFT, and direct customer entry through the Internet. The master file concept has been largely displaced by database management systems, and static reporting replaced or augmented by ad-hoc reporting and direct inquiry, including downloading of data by customers. The ubiquity of the Internet and the Personal Computer have been the driving force in the transformation of Data Processing to the more global concept of Data Management Systems.

Characteristics of Database

The database approach has some very characteristic features which are discussed in detail

below:
6
1.5.1 Concurrent Use
A database system allows several users to access the database concurrently. Answering different questions from different users with the same (base) data is a central aspect of an information system. Such concurrent use of data increases the economy of a system.
An example for concurrent use is the travel database of a bigger travel agency. The employees of different branches can access the database concurrently and book journeys for their clients. Each travel agent sees on his interface if there are still seats available for a specific journey or if it is already fully booked.
1.5.2 Structured and Described Data
A fundamental feature of the database approach is that the database systems does not only contain the data but also the complete definition and description of these data. These descriptions are basically details about the extent, the structure, the type and the format of all data and, additionally, the relationship between the data. This kind of stored data is called metadata ("data about data").
1.5.3 Separation of Data and Applications
As described in the feature structured data the structure of a database is described through *metadata* which is also stored in the database. An application software does not need any knowledge about the physical data storage like encoding, format, storage place, etc. It only communicates with the management system f a database (DBMS) via a standardised interface with the help of a standardised language like SQL. The access to the data and the metadata is entirely done by the DBMS. In this way all the applications can be totally seperated from the data. Therefore database internal reorganisations or improvement of efficiency do not have any influence on the application software.
1.5.4 Data Integrity
Data integrity is a byword for the quality and the reliability of the data of a database system. In a broader sense data integrity includes also the protection of the database from unauthorised access (confidentiality) and unauthorised changes. Data reflect facts of the real world. database.
1.5.5 Transactions
A transaction is a bundle of actions which are done within a database to bring it from one
7
consistent state to a new consistent state. In between the data are inevitable inconsistent. A transaction is atomic what means that it cannot be divided up any further. Within a transaction all or none of the actions need to be carried out. Doing only a part of the actions would lead to an inconsistent database state. One example of a transaction is the transfer of an amount of money from one bank account to another. The debit of the money from one account and the credit of it to another account makes together a consistent transaction. This transaction is also atomic. The debit or credit alone would both lead to an inconsistent state. After finishing the transaction (debit and credit) the changes to both accounts become persistent and the one who gave the money has now less money on his account while the receiver has now a higher balance.

1.5.6 Data Persistence

Data persistence means that in a DBMS all data is maintained as long as it is not deleted explicitly. The life span of data needs to be determined directly or indirectly be the user and must not be dependent on system features. Additionally data once stored in a database must not be lost. Changes of a database which are done by a transaction are persistent. When a transaction is finished even a system crash cannot put the data in danger.

1.6 Advantages and Disadvantages of a DBMS

Using a DBMS to manage data has many advantages:

Data independence: Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.

Efficient data access: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.

Data integrity and security: If data is always accessed through the DBMS, the DBMS can enforce integrity constraints on the data. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, the DBMS can enforce *access controls* that govern what data is visible to different classes of users.

Data administration: When several users share the data, centralizing the administration of data can offer significant improvements. Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and finetuning the storage of the data to make retrieval efficient.

Concurrent access and crash recovery: A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.

Reduced application development time: Clearly, the DBMS supports many important functions that are common to many applications accessing data stored in the DBMS. This, in conjunction with the high-level interface to the data, facilitates quick development of applications. Such applications are also likely to be more robust than applications developed from scratch because many important
tasks are handled by the DBMS instead of being implemented by the application. Given all these advantages, is there ever a reason *not* to use a DBMS? A DBMS is a complex piece of software, optimized for certain kinds of workloads (e.g., answering complex queries or handling many concurrent requests), and its performance may not be adequate for certain specialized applications. Examples include applications with tight real-time constraints or applications with just a few well-designed critical operations for which efficient custom code must be written. Another reason for not using a DBMS is that an application may need to manipulate the data in ways not supported by the query language. In such a situation, the abstract view of the data presented by the DBMS does

not match the application's needs, and actually gets in the way. As an example, relational databases do not support flexible analysis of text data (although vendors are now extending their products in this direction). If specialized performance or data manipulation requirements are central to an application, the application may choose not to use a DBMS, especially if the added benefits of a DBMS (e.g., flexible querying, security, concurrent access, and crash recovery) are not required. In most situations calling for large-scale data management, however, DBMSs have become an indispensable tool.

9

Disadvantages of a DBMS

Danger of a Overkill: For small and simple applications for single users a database system is often not advisable.

Complexity: A database system creates additional complexity and requirements. The supply and operation of a database management system with several users and databases is quite costly and demanding.

Qualified Personnel: The professional operation of a database system requires appropriately trained staff. Without a qualified database administrator nothing will work for long.

Costs: Through the use of a database system new costs are generated for the system itselfs but also for additional hardware and the more complex handling of the system.

Lower Efficiency: A database system is a multi-use software which is often less efficient than specialised software which is produced and optimised exactly for one problem.

Data Dictionary

It holds detailed information about the different structures and data types, the details of the logical structures that are mapped into the different structure, details of relationship between data items, details of all users privileges and access rights and performance of resource with details.

TYPES OF RELATIONSHIPS

There are three types of relationships between the tables. The type of relationship that is created depends on how the related columns are defined:

One-to-one relationship (1:1)

A pair of tables bears a one-to-one relationship when a single record in the first table is related to only one record in the second table, and a single record in the second table is related to only one record in the first table.

One-to-Many Relationships (1:M)

A one-to-many relationship exists between a pair of tables when a single record in the first table can be related to one or more records in the second table, but a single record in the second table can be related to only one record in the first table.

Many-to-Many Relationships (M:M)

A pair of tables bears a many-to-many relationship when a single record in the first

table can be related to one or more records in the second table and a single record in the second table can be related to one or more records in the first table.

Data Independence

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

1. **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), or to reduce the database (by removing a record type or data item). In the latter case, external schemas that refer only to the remaining data should not be affected. Only the view definition and the mappings need be changed in a DBMS that supports logical data independence. Application programs that reference the external schema constructs must work as before, after the conceptual schema undergoes a logical reorganization. Changes to constraints can be applied also to the conceptual schema without affecting the external schemas or application programs.

2. **Physical data independence** is the capacity to change the internal schema without having to change the conceptual (or external) schemas. Changes to the internal schema may be needed because some physical files had to be reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema.

Whenever we have a multiple-level DBMS, its catalog must be expanded to include information on how to map requests and data among the various levels. The DBMS uses additional software to accomplish these mappings by referring to the mapping information in the catalog. Data independence is accomplished 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. Hence, application programs referring to the higher-level schema need not be changed.

The three-schema architecture can make it easier to achieve true data independence, both physical and logical. However, the two levels of mappings create an overhead during compilation or execution of a query or program, leading to inefficiencies in the DBMS. Because of this, few DBMSs have implemented the full three-schema architecture

INTRODUCTION

- DBMS stands for Database Management System.
- DBMS is a software system for creating, organizing and managing the database.
- It provides an environment to the user to perform operations on the database for creation, insertion, deletion, updating and retrieval of data.

What is Data ?

- A collection of raw facts and figures.
- Raw material that can be processed by any computing machine.
- A collection of facts from which conclusions may be drawn.
- Data can be represented in the form of: numbers and words which can be stored in computer's language.
  i.e. Asif khan, Asad ,001,

What is Information?

- Systematic and meaningful form of data.
- Knowledge acquired through study or experience.
- Information helps human beings in their decision making.

Database

- A safekeeping of logically related and similar data.
- An organized collection of related information so that it can easily be accessed, managed and updated.

  E.g.:

  - Dictionary
  - Airline Database
  - Student Database
  - Library
  - Railways Timetable
  - YouTube

What is DBMS ?

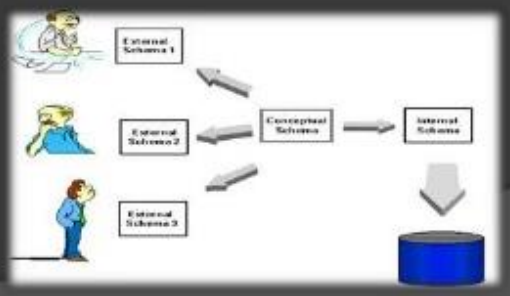
- A set of programs to access the interrelated data.
- DBMS contains information about a particular enterprise.
- Computerized record keeping system.
- Provides convenient environment to user to perform operations:

  -Creation, Insertion, Deletion,

  Updating & Retrieval of information.

Database Users

- Database users and user interfaces
  - Naive Users
  - Application Programmers
  - Sophisticated Users
  - Specialized Users

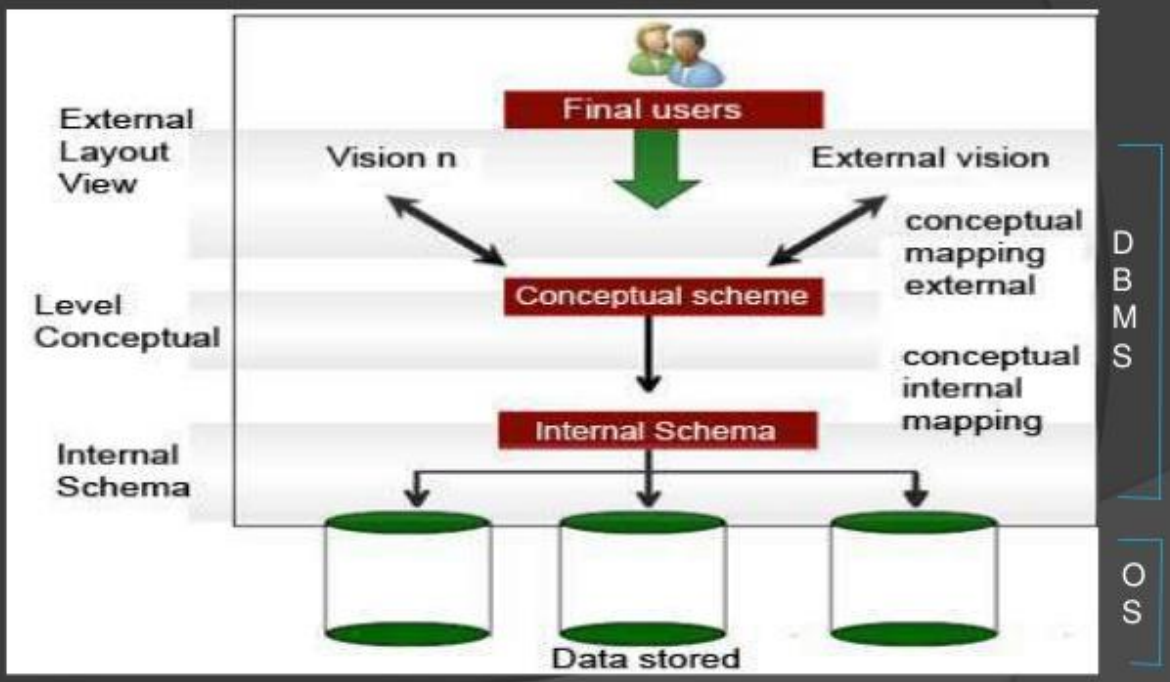


Database Administrator (DBA)

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance

- DBA manage all level of DBMS model

Three Levels of Architecture



External View

- Highest or Top level of data abstraction (No knowledge of DBMS S/W and H/W or physical storage).
- This level is concerned with the user.
- Each external schema describes the part of the database that a particular user is interested in and hides the rest of the database from user.
- There can be n number of external views for database where n is the number of users.
- For example, a accounts department may only be interested in the student fee details. It would not be expected to have any interest in the personal information about students.
- All database users work on external level of Dbms .

Conceptual View

- This level is in between the user level and physical storage view.
- There is only one conceptual view for single database.
- It hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.

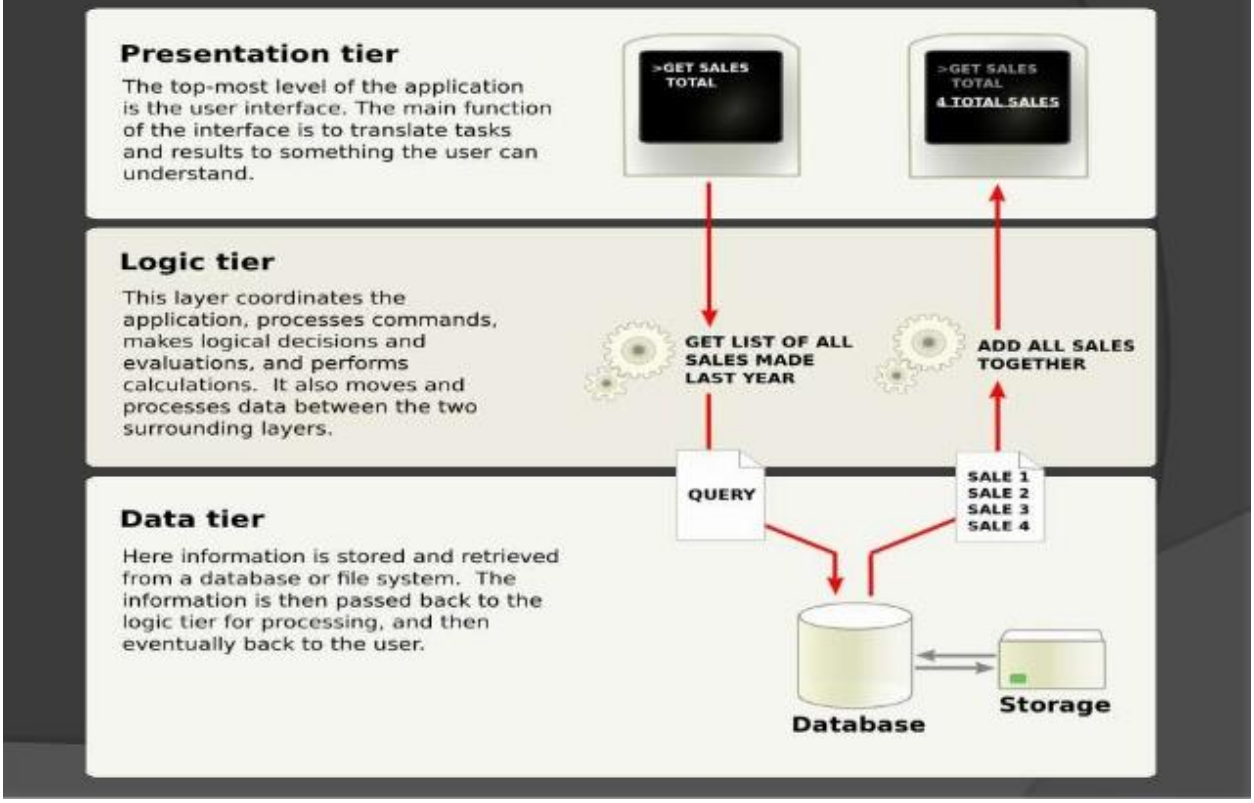
Internal View

- It is the lowest level of data abstraction. (it has the knowledge about s/w and h/w)
- At this level, it keeps the information about the actual representation of the entire database i.e. the actual storage of the data on the disk in the form of records or blocks.
- It is close to the physical storage method.
- The internal view is the view that tells us what data is stored in the database and how. At least the following aspects are considered at this level: Storage allocation, Access paths etc.
- The internal view does not deal with the physical devices directly. Instead it views a physical device as a collection of physical pages and allocates space in terms of logical pages.

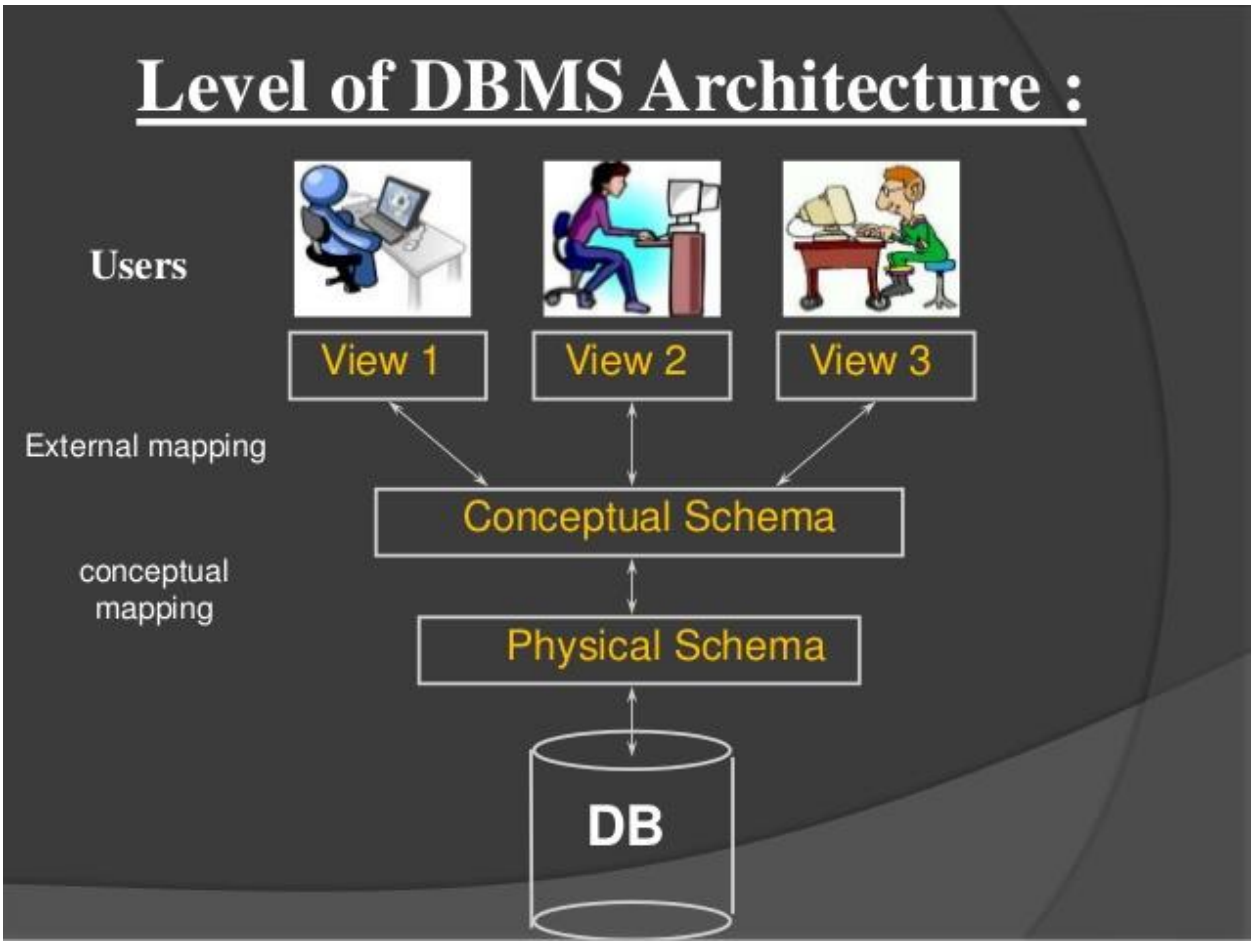
Three Levels of Architecture (cont…)

- Internal/physical level: Shows how data are stored inside the system. It is the closest level to the physical storage. This level talks about database implementation and describes such things as file organization and access paths of indexes, data compression and encryption techniques, and record placement
- Conceptual/logical level: Deals with the modeling of the whole database. The conceptual schema of database is defined in this level
- External level: This level models a user oriented description of part of the database. The views for individual users are defined by means of external schemas in this level

Working of three level architecture



Level of DBMS Architecture :



Example: University Database

- External Schema (View 1): *Course_info(cid:string,cname:string)*
- External Schema (View 2): *student_info(cid:string, name:string)*
- Conceptual schema:
  - *Students(sid: string, name: string, login: string, age: integer)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(Eid:string, cid:string, grade:string)*
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.

Example: employee database

External view 1 (C++)

```
DCL 1 EMPP,
      2 EMP# CHAR(6)
      2 SAL FIXEDBIN(31)
```

External view 2 (COBOL)

```
01  EMPC.
      02 EMPNO PIC X(6).
      02 DEPTNO PIC X(4).
```

Conceptual

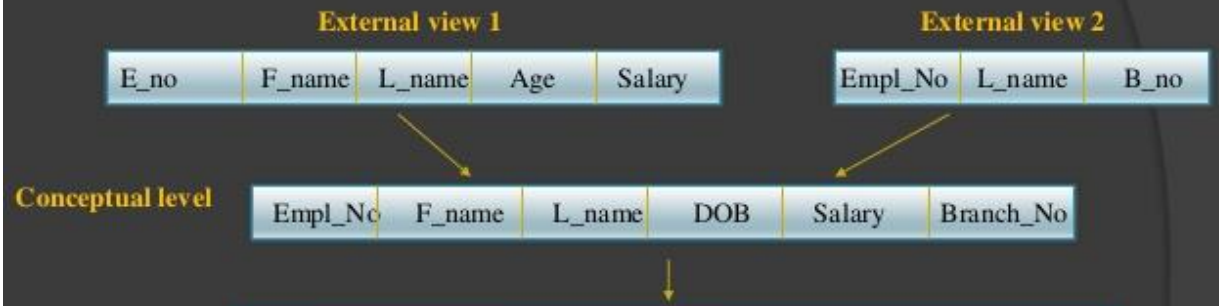
```
EMPLOYEE
    EMPLOYEE_NUMBER     CHARACTER(6)
    DEPARTMENT_NUMBER  CHARACTER(6)
    SALARY                    DECIMAL(5)
```

Internal

```
STORED_EMP  BYTES=20
  PREFIX      BYTE=6 , OFFSET=0
  EMP#        BYTE=6, OFFSET=6, INDEX=EMPX
  DEPT#       BYTES=4, OFFSET=12
  PAY         BYTES= 4, ALIGN= FULLWORD,OFFSET=16
```

Three Levels of Architecture

Syntax Example:



Conceptual level

Empl_No	F_name	L_name	DOB	Salary	Branch_No

Internal level

```
struct EMPLOYEE {
    int Empl_No;
    int Branch_No;
    char F_name [15];
    char L_name [15];
    struct date Date_of_Birth;
    float Salary;
    struct EMPLOYEE *next;      //pointer to next employee record
    };   index Empl_No; index Branch_No;      //define indexes for
employees
```

Three Level Architecture Objectives

- Each user should be able to access the same data but have a different customize view of the data.
- User should not have to deal directly with physical database storage detail.
- The DBA should be able to change the database storage structure without affecting the users views.

Three Level Architecture Objectives

- The internal structure of the database should be unaffected by changes to the physical aspects of storage.

- The DBA should be able to change the conceptual structure of the database without affecting all users.

Mapping

- Mapping is the key for providing data independence
- Data independence is the capacity to change the schema at one level without having to change the schema at the next higher level
- Two types of data independence are
  - Logical data independence
  - Physical data independence

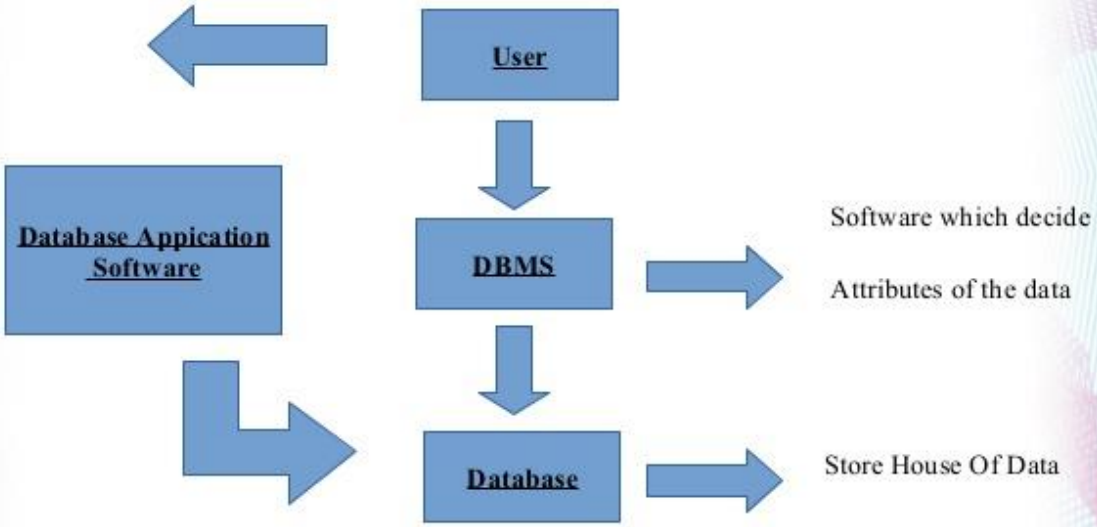
Mapping - Data Independence

- Logical data independence (provided by external/ conceptual mapping)
  - Ability to modify conceptual schema without changing
    - External views
    - Application programs
  - Changes to conceptual schema may be necessary
    - Whenever the logical structure of the database changes
      - **Due to changed objectives**
  - Examples
    - Adding a data item to schema
    - Adding price of a part to PART table
    - Adding PROJECT table to the SUPPLIER-PARTS database

Mapping - Data Independence

- Physical data independence (provided by conceptual/internal mapping)
  - Ability to modify internal or physical schema without changing
    - Conceptual or view level schema
    - Application programs
  - Changes to physical schema may be necessary to
    - Improve performance of retrieval or update
- Achieving logical data independence is more difficult than physical data independence
  - Because application programs heavily rely on the logical structure of the data they access

Database Management System



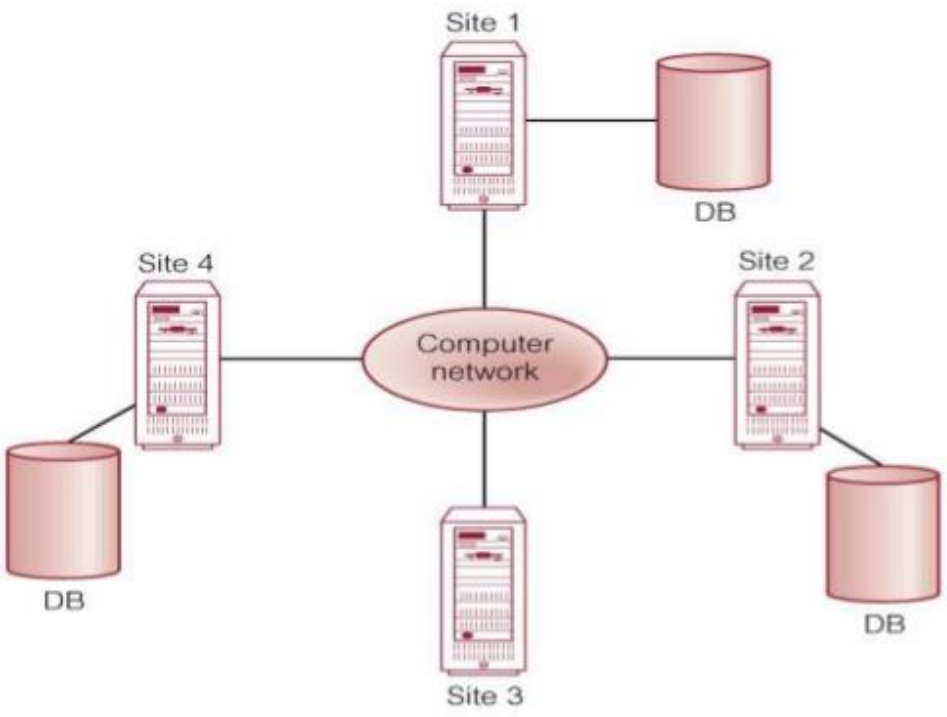
Distributed Database

- Database:- Logical interrelated collection of shared data, along with description of data, physically distributed over a computer network.

What is Distributed Database?

- A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network.

- A distributed database management system (DDBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users


Site 1
DB
Site 4
Site 2
Computer network
DB
DB
Site 3

- A DDBMS mainly classified into two types:
  - Homogeneous Distributed database management systems
  - Heterogeneous Distributed database management systems

Characteristics

- All sites are interconnected.
- Fragments can be replicated.
- Logically related shared data can be collected.
- Data at each and every site is controlled by the DBMS.
- Each Distributed Database Management System takes part in at least one global application.

Functionality

- Security
- Keeping track of data
- Replicated data management
- System catalog management
- Distributed transaction management
- Distributed database recovery

Advantages

- Less danger of a single-point failure. When one of the computers fails, the workload is picked up by other workstations.
- Data are also distributed at multiple sites.
- The end user is able to access any available copy of the data, and an end user's request is processed by any processor at the data location.

Advantages (contd..)

- Improved communications. Because local sites are smaller and located closer to customers.
- Reduced operating costs. It is more cost-effective to add workstations to a network than to update a mainframe system.
- Faster data access, faster data processing.
- A distributed database system spreads out the systems workload by processing data at several sites.

Disadvantages

- Complexity of management and control.
- Applications must recognize data location, and they must be able to stitch together data from various sites.
- Security.

Disadvantages (contd..)

- Increased storage and infrastructure requirements.
- Multiple copies of data has to be at different sites, thus an additional disk storage space will be required.
- The probability of security lapses increases when data are located at multiple sites.