Chapter One

Introduction to Database concepts

Data, information, information System

Data: is a collection of raw facts.

Information: is a processed data in the form that is meaningful to the user.

Information System is a system that:

- Receives data and instruction
- Processes the data as per the instruction
- Produces output
- Stores data/information for future use

Information System and Organization, database system

Information System doesn't exist without organization. That is, organization of data is necessary if data is voluminous. Information System is a support system for the organizational activity to achieve a certain goal.

A database system is basically a computerized record keeping system. Users of the database can perform a variety of operations. Such as:

- Adding new data to empty file
- Adding new data to existing file
- Retrieving data from existing file
- Modifying data to existing file
- Deleting data from existing file
- Searching for target information

Data handling approaches

Data management passes through the different levels of development along with the development in technology and services. These levels could be best described by categorizing the levels into three levels or types of development/approach. Even though there is an advantage and a problem overcome at each new data handling approach/level, all methods or

approaches of data handling are in use to some extent. The major three approaches/levels are discussed as follows:

1. Manual Approach

In the manual data handling approach, data storage and retrieval follows the primitive and traditional way of data/information handling where cards and paper are used for the purpose. Typing the data on paper and put in a file cabinet. The data storage and retrieval will be performed using human labour. This approach works well if the number of items to be stored is small.

Limitations of the Manual approach

- Prone to error
- Data loss: Due to damaged papers or unable to locate it.
- Redundancy: Multiple copies of the same data within the organization.
- Inconsistency: Modifications are not reflected on all multiple copies
- Difficult to update, retrieve, integrate
- You have the data but it is difficult to compile the information
- Limited to small size information
- Cross referencing is difficult

An alternative approach of data handling is a computerized way of dealing with the information. The computerized approach could also be either decentralized or centralized based on where the data resides in the system.

2. File based Approach

After the introduction of computer for data processing to the business community, the need to use the device for data storage and processing increase. File based data handling approaches were an early attempt to computerize the manual filing system. There were, and still are, several computer applications with file based processing used for the purpose of data handling. It is a collection of application programs that performs services for the end users. In such systems, every application program that provides service to end users define and manage its own data.

Such systems have number of programs for each of the different applications in the organization. And this approach is the decentralized computerized data handling method.

Limitations of the File Based approach

As business application become more complex and demanding more flexible and reliable data handling methods, the shortcomings of the file based system became evident. These shortcomings include, but not limited to:

- **Separation/Isolation of data**

When data is isolated in separate files, it is difficult to access data that should be available. This is because; there is no concept of relationship between files. Therefore, we need to create a temporary file for the participating files.

- **Duplication of data (Redundancy)**

This is concerning with storage of similar information in multiple files.

The following are some of the disadvantage of redundancy:

- ✓ It costs time and money to enter the data
- ✓ It takes up additional storage space (memory space)
- ✓ Inconsistency: this is loss of data integrity. For instance, if modification in the child table is unable to be reflected on the parent table.

- **Data Dependence**

Changes to an existing structure are difficult to make. Example: change in the size of Student Name (from 20 characters to 30 characters) requires a new program to convert student file to a new format. The new program opens original student file, open a temporary file, read records from original student file and write to the temporary file, delete the original student file and finally rename the temporary file as student file. It is time consuming and Prone to error.

- **Incompatible file formats**

The structure of file is dependent on the application programs. Incompatibility of files makes them difficult to process jointly. Example: consider two files with in the same enterprise but in different departments, or in different branches: If the first file is constructed using COBOL and the second file is written using C++, then there will be a problem of integrity.

3. Database Approach

What is a Database?

A database is a collection of related data in an organized way. Most of the time, organization is in tabular form. E.g. book database

Call no	Title	Author	Publisher	No of copies
QA46	Introduction to database	Bahiru	Addison Wesley	15

The organization of the database becomes necessary when the data is voluminous. Otherwise, managing data will be very difficult.

E.g. A Manufacturing Company with product data

A Bank with account data

A Hospital with patients

A University with Students

A government with planning data

What is a database system?

It is a computerized record keeping system, which stores related data in an organized way. The overall purpose of a database system is to store information and to allow users to add, delete, retrieve, search, query and update that information upon request. The information concerned can be anything that is deemed to be of significance to the individual or organization the system is intended to serve. That is, needed to assist in the general process of running the business of that individual or organization.

Thus, in database approach:

- Database is just a computerized record keeping system or a kind of electronic filing cabinet.
- Database is a repository for collection of computerized data files.
- Database is a shared collection of logically related data designed to meet the information needs of an organization. Since it is a shared corporate resource, the database is integrated with minimum amount of or no duplication.

- Database is a collection of logically related data where these logically related data comprises: Entities, Attributes, Relationships, and business rules of an organization's information.
- In addition to containing data required by an organization, database also contains a description of the data which called as “Metadata” or “Data Dictionary” or “Systems Catalogue” or “Data about Data”.
- Since a database contains information about the data (metadata), it is called a self descriptive collection on integrated records.
- The purpose of a database is to store information and to allow users to retrieve and update that information on demand.
- Database is designed once and used simultaneously by many users.
- Unlike the traditional file based approach in database approach there is program data independence. That is the separation of the data definition from the application. Thus the application is not affected by changes made in the data structure and file organization.
- Each database application will perform the combination of: Creating database, Reading, Updating and Deleting data.

The advantages of a database approach over the traditional and paper-based methods of record keeping will include the following:

Compactness: no need for possibly voluminous paper files.

Speed: the machine can retrieve and change data faster than a human can. In particular, ad hoc, spur-of-the-moment queries

(“Do we have more red screws than blue ones?”) can be answered quickly without any need for time consuming manual or visual searches.

Accuracy: timely, accurate and up-to-date information is available on demand at any time.

The foregoing benefits apply with even more force in a multi-user environment where the database is likely to be much larger and much more complex than in the single user case. In a multi-user environment the database system provides the enterprise with centralized control of its data. The centralized approach has the following advantages:

Data can be shared: two or more users can access and use same data instead of storing data in redundant manner for each user.

Redundancy can be reduced: In non database or non centralized systems each application or department keeps its own private files. The files may hold common data elements that exist as part of the enterprises data. This will lead to considerable redundancy in stored data, with resultant waste in storage space. For example, a personnel application and an education records application might both own a file that includes department information for employees. Note that, this is not to say we should eliminate all redundancies. Sometimes there are sound reasons for maintaining several copies of the same data.

Inconsistency can (to some extent) avoided: If there are a number of files which store similar data elements among other sorts of data then when a change is made to a particular data (among the common ones) this change need to be done throughout the system where there is such data stored. This is not, often, the case. Some of the data might be updated and others left as they are which results in inconsistent information about the same phenomena.

Standards can be enforced: Standardizing data representation is particularly desirable as an aid to data interchange or migration of data between systems. Likewise, data naming and documentation standards are also very desirable as they facilitate data sharing and understandability.

Security restrictions can be applied: Since the data is stored in one place/area all accesses to the data can be regulated by the system through some defined rules built into the system. The system ensures that the only means of access to the database is through proper channels. Different rules can be established for each type of access (retrieve, insert, delete, etc.) to each of information to the database.

Integrity can be maintained The problem of integrity is the problem of ensuring the data in the database is accurate. Inconsistency between two entries that represent the same “fact” is an example of lack of integrity. It is more serious in a multi-user environment where one user may enter bad data and other users may go on working on the updated data as if it were a correct one.

Conflicting requirements can be balanced: knowing the overall requirements of the enterprise the Database Administrator (DBA) can structure the system so as to provide an overall service that is best for the enterprise. For example, a representation can be chosen for the data in storage that gives fast access for the most important applications (possibly at the cost of poorer performance for certain other applications).

Transaction support can be provided: basic demands of any transaction support systems are implanted in a full scale DBMS.

Improved decision support: the database will provide information useful for decision making.

Less labour: unlike the other data handling methods, data maintenance will not demand much resource.

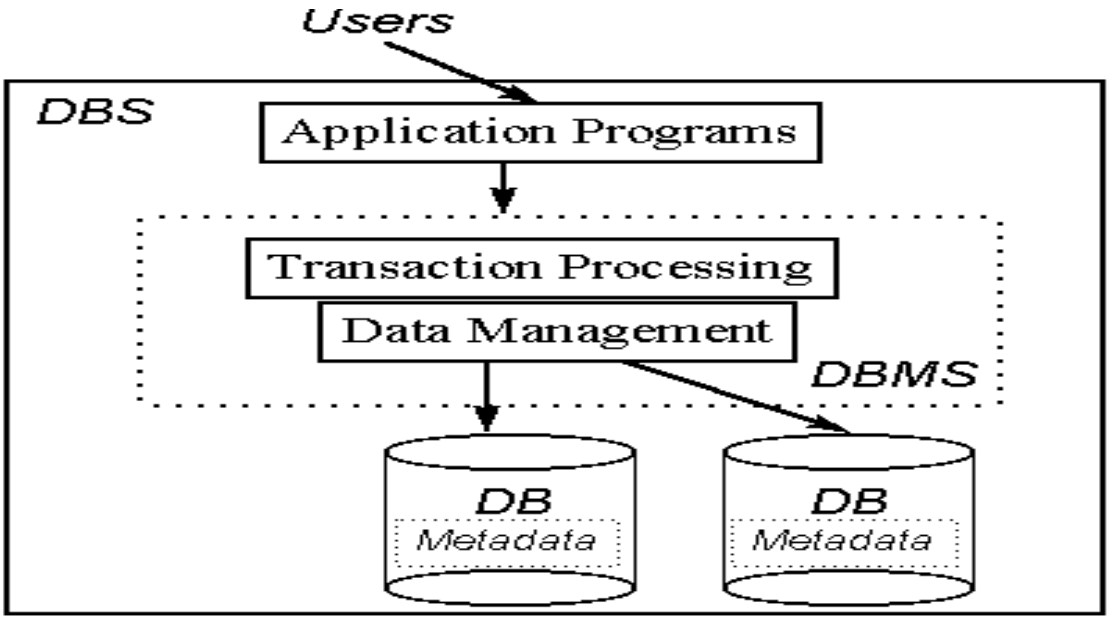
Centralized information control: since relevant data in the organization will be stored at one repository, it can be controlled and managed at the central level.

Limitations and risk of Database Approach

- Introduction of new professional and specialized personnel.
- Complexity in designing and managing data
- The cost and risk during conversion from the old to the new system
- High cost to be incurred to develop and maintain the system
- Complex backup and recovery services from the users perspective
- Reduced performance due to centralization and data independency
- High impact on the system when failure occurs to the central system.

Note: Database System (DBS) contains:

The Database + The DBMS + Application Programs (what users interact with)



Components of a Database System

A database system involves four major components, namely, data, hardware, software and users and designers of database. A brief discussion will follow on each of these components.

Data: The actual data stored in the database system may be stored as a single database or distributed in many distinct files and treated as one. Is the system a single-user or multi-user one? How are we going to achieve the utmost possible performance concerning the data storage and maintenance? What other benefits or drawbacks do we expect as the result of placement or structure of the database?. These and similar issues might be concerned with the way the data stored in the system.

Hardware: This portion of the system consists of secondary storage media (disks, tapes and optical media) that are used to hold the stored data and associated device controllers (hard disk controller, etc.); and the processor(s) and associated main memory that are used to support the execution of the database system software.

Software: This is the software, Database Management System (DBMS) that is responsible for the overall management of communications between the user and the database. It is found between the data and the users, which, in other words, means the data is entirely covered or shielded by the DBMS software. The DBMS provides facilities for operating on the database. This is the most

important software component in the overall system that allows the user to interact with the data.

Users and Designers of Database: As people are one of the components in DBS environment, there are group of roles played by different stakeholders of the designing and operation of a database system.

1. **Database Administrator (DBA)**
   - Responsible to oversee, control and manage the database resources (the database itself, the DBMS and other related software)
   - Authorizing access to the database
   - Coordinating and monitoring the use of the database
   - Responsible for determining and acquiring hardware and software resources
   - Accountable for problems like poor security, poor performance of the system
   - Involves in all steps of database development

We can have further classifications of this role in big organizations having huge amount of data and user requirement.

- **Data Administrator (DA)**: is responsible on management of data resources. Involves in database planning, development, maintenance of standards policies and procedures at the conceptual and logical design phases.
- **Database Administrator (DBA):** is more technically oriented role. Responsible for the physical realization of the database. Involves in physical design, implementation, security and integrity control of the database. Database Administrator a person that is responsible for all technical operations or details of the database system. The user that controls the enterprises data resource.

The functions of the DBA include the following.

- **Defining the conceptual schema:** Will directly participate or help on the process of identifying the content of the database, i.e., what information is to be held in the database and create the corresponding conceptual schema using the conceptual DDL.

- **Defining the internal schema:** The DBA must also decide how the data is to be represented in the stored database and then create the corresponding storage structure definition (the internal schema) using the internal DDL (including associated mapping between the internal and conceptual schema).
- **Liaising with users:** By communicating with users the DBA will ensure that the data they require is available, and to write (or help users write) the necessary external schemas using the applicable external DDL. Other functions include consulting on application design, providing technical education, assisting with problem determination and resolution, and similar system related professional services.
- **Defining security and integrity rules:** Since security and integrity rules are part of the conceptual schema, the conceptual DDL should include facilities for specifying such rules.
- **Defining backup and recovery procedures:** In the event of damage to any portion of a database, caused by human error or failure in the hardware or operating system, it is essential to be able to repair the data concerned with the minimum of delay and with as little effect as possible on the rest of the system. The DBA should define and implement appropriate backup and recovery scheme.
- **Monitoring performance and responding to changing requirements:** Periodic performance analysis should be done by the DBA and based on the results obtained propose for improved systems and or do the modifications on the existing data definitions

2. Database Designer (DBD)

- Identifies the data to be stored and choose the appropriate structures to represent and store the data.
- Should understand the user requirement and should choose how the user views the database.
- Involve on the design phase before the implementation of the database system.

We have two distinctions of database designers, one involving in the logical and conceptual design and another involving in physical design.

- **Logical and Conceptual DBD**
  - Identifies data (entity, attributes and relationship) relevant to the organization
  - Identifies constraints on each data
  - Understand data and business rules in the organization
  - Sees the database independent of any data model at conceptual level and consider one specific data model at logical design phase.
- **Physical DBD**
  - Take logical design specification as input and decide how it should be physically realized.
  - Map the logical data model on the specified DBMS with respect to tables and integrity constraints. (DBMS dependent designing)
  - Select specific storage structure and access path to the database
  - Design security measures required on the database

3. Application Programmer and Systems Analyst

- System analyst determines the user requirement and how the user wants to view the database.
- The application programmer implements these specifications as programs; code, test, debug, document and maintain the application program.
- Determines the interface on how to retrieve, insert, update and delete data in the database.
- The application could use any high-level programming language according to the availability, the facility and the required service.

Application programmers who are responsible for writing application programs that use the database using some programming language such as COBOL, Pascal, or a programming language built-in to the DBMS.

4. **End-users:** These are those people who are engaged on processing different types of operations on the database system. Users are workers, whose job requires accessing the database frequently for various purpose. There are different group of users in this category.

- **Naïve Users:**
  - ✓ Sizable proportion of users
  - ✓ Unaware of the DBMS
  - ✓ Only access the database based on their access level and demand
  - ✓ Use standard and pre-specified types of queries.
- **Sophisticated Users**
  - ✓ Familiar with the structure of the Database and facilities of the DBMS.
  - ✓ Have complex requirements
  - ✓ Have higher level queries
  - ✓ Are most of the time engineers, scientists, business analysts, etc
- **Casual Users**
  - ✓ Users who access the database occasionally.
  - ✓ Need different information from the database each time.
  - ✓ Use sophisticated database queries to satisfy their needs.
  - ✓ Are most of the time middle to high level managers.

Generally**,** End users are those that interact with the system from online workstations or terminals that use an application program developed by application programmers or those that query the system through an interface provided by the DBMS.

These users can be again classified as “Actors on the Scene” and “Workers Behind the Scene”.

Actors On the Scene:

- Data Administrator
- Database Administrator
- Database Designer
- End Users

Workers Behind the Scene

- **DBMS** designers and implementers: who design and implement different DBMS software.
- **Tool Developers**: experts who develop software packages that facilitates database system designing and use. Prototype, simulation, code generator developers could be an example. Independent software vendors could also be categorized in this group.
- **Operators and Maintenance Personnel**: system administrators who are responsible for actually running and maintaining the hardware and software of the database system and the information technology facilities.

Database Management System (DBMS)

Database Management System (DBMS) is the tool for creating and managing the large amounts of data efficiently and allowing it to persist for a long period of time. Hence DBMS is a general-purpose software that facilities the processes of defining, constructing, manipulating, and sharing database.

- Defining: involves specifying data types, structure and constraints.
- Constructing: is the process of storing the data into a storage media.
- Manipulating: is retrieving and updating data from and into the storage.
- Sharing: allows multiple users to access data.

A DBMS is software that enables users to define, create, maintain and control access to the database. Example: Ms Access, FoxPro, SQL Server, MySQL, Oracle.

The phrase “Database System” is used to colloquially refer to database and database management system (DBMS).

- Database Management System (DBMS) is a Software package used for providing EFFICIENT, CONVENIENT and SAFE MULTI-USER (many people/programs accessing same database, or even same data, simultaneously) storage of and access to MASSIVE amounts of PERSISTENT (data outlives programs that operate on it) data.
- A DBMS also provides a systematic method for creating, updating, storing, retrieving data in a database.

- DBMS also provides the service of controlling data access, enforcing data integrity, managing concurrency control, and recovery.

Having this in mind, a full scale DBMS should at least have the following services to provide to the user.

1. Data **storage**, **retrieval** and **update** in the database
2. A user accessible **catalogue**
3. **Transaction support service**: ALL or NONE transaction, which minimize data inconsistency.
4. **Concurrency Control Services**: access and update on the database by different users simultaneously should be implemented correctly.
5. **Recovery Services**: a mechanism for recovering the database after a failure must be available.
6. **Authorization Services (Security):** must support the implementation of access and authorization service to database administrator and users.
7. **Support for Data Communication**: should provide the facility to integrate with data transfer software or data communication managers.
8. **Integrity Services**: rules about data and the change that took place on the data, correctness and consistency of stored data, and quality of data based on business constraints.
9. Services to promote **data independency** between the data and the application

Components of DBMS Environment

A DBMS is software package used to design, manage, and maintain databases. Each DBMS should have facilities to define the database, manipulate the content of the database and control the database. These facilities will help the designer, the user as well as the database administrator to discharge their responsibility in designing, using and managing the database. It provides the following facilities:

- **Data Definition Language (DDL):**
  - Language used to define each data element required by the organization.
  - Commands for setting up schema or the intension of database

- These commands are used to setup a database, create, delete and alter table with the facility of handling constraints
- Allows DBA or user to describe and name entitles, attributes and relationships required for the application.
- Specification notation for defining the database schema

– **Data Manipulation Language (DML):**

- Is a core command used by end-users and programmers to store, delete, and upate the data in the database.
- Provides basic data manipulation operations on data held in the database.
- Language for manipulating the data organized by the appropriate data model

– **Data Query Language (DQL):**

- Language for accessing or retrieving the data organized by the appropriate data model.
- Since the required data or Query by the user will be extracted using this type of language, it is also called "Query Language"
  - ✓ **Procedural DQL:** user specifies what data is required and how to get the data.
  - ✓ **Non-Procedural DQL:** user specifies what data is required but not how it is to be retrieved

– **Data Dictionary (DD):**

- Due to the fact that a database is a self describing system, this tool, Data Dictionary, is used to store and organize information about the data stored in the database.

– **Data Control Language (DCL):**

- Database is a shared resource that demands control of data access and usage. The database administrator should have the facility to control the overall operation of the system.
- Data Control Languages are commands that will help the Database Administrator to control the database.
- The commands include grant or revoke privileges to access the database or particular object within the database and to store or remove database transactions

Database System Development Life Cycle

As it is one component in most information system development tasks, there are several steps in developing a database system. Here more emphasis is given to the design phases of the database system development life cycle. The major steps in database system development are;

1. **Planning**: That is identifying information gap in an organization and propose a database solution to solve the problem.
2. **Analysis**: That concentrates more on fact finding about the problem or the opportunity. Feasibility analysis, requirement determination and structuring, and selection of best design method are also performed at this phase.
3. **Design**: In database system development more emphasis is given to this phase. The phase is further divided into three sub-phases.
   A. **Conceptual Design**: Concise description of the data, data type, relationship between data and constraints on the data.
      - There is no implementation or physical detail consideration.
      - Used to elicit and structure all information requirements.
   B. **Logical Design**: A higher level conceptual abstraction with selected specific database model to implement the data structure.
      - It is particular DBMS **independent** and with no other physical considerations.
   C. **Physical Design**: Physical implementation of the upper level design of the database with respect to internal storage and file structure of the database for the selected DBMS.
      - To develop all technology and organizational specification.
4. **Implementation**: The testing and deployment of the designed database for use.
5. **Operation and Support**: administering and maintaining the operation of the database system and providing support to users.

Basic Conceptes

- **Database Design:** The activity of specifying the schema of a database in a given data model

- **Database Schema:** The structure of a database that:
  - Captures data types, relationships and constraints in data
  - Is independent of any application program
  - Changes infrequently
- **Data Model:**
  - A set of primitives for defining the structure of a database.
  - A set of operations for specifying retrieval and updates on a database
  - Examples: Relational, Hierarchical, Networked, Object-Oriented
- **Database Instance or State:** The actual data contained in a database at a given time.

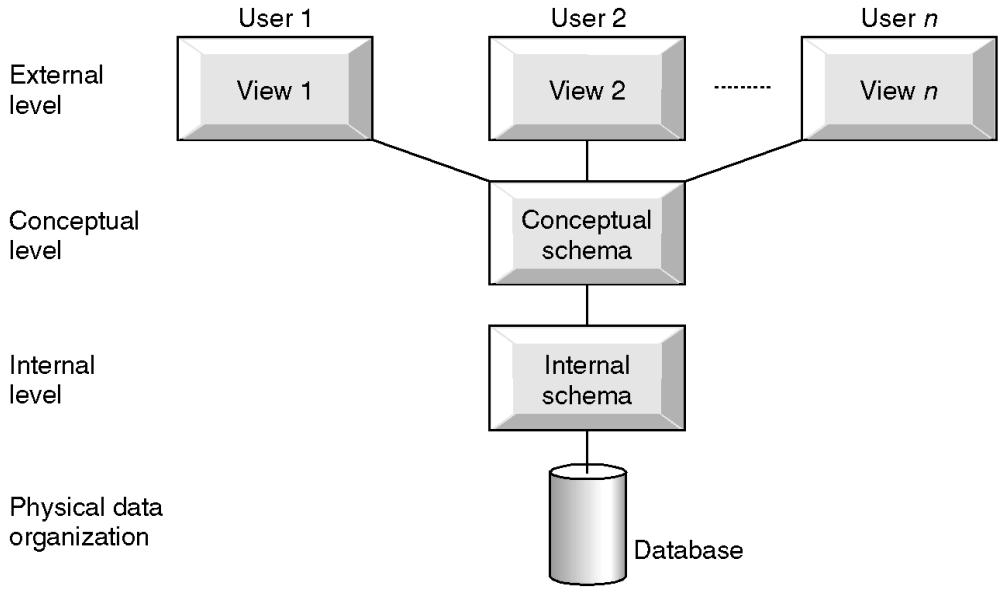
Database Systems Architecture

There may be several types of architectures of database systems. However, the following architecture (ANSI/SPARC) is applicable to most modern database systems. **External level**, **Conceptual level** and **Internal level**.

- All users should be able to access same data. This is important since the database is having a shared data feature where all the data is stored in one location and all users will have their own customized way of interacting with the data.
- A user's view is unaffected or immune to changes made in other views. Since the requirement of one user is independent of the other, a change made in one user's view should not affect other users.
- Users should not need to know physical database storage details. As there are naïve users of the system, hardware level or physical details should be a black-box for such users.
- DBA should be able to change database storage structures without affecting the users' views. A change in file organization, access method should not affect the structure of the data which in turn will have no effect on the users.
- Internal structure of database should be unaffected by changes to physical aspects of storage.
- DBA should be able to change conceptual structure of database without affecting all users. In any database system, the DBA will have the privilege to change the structure of the

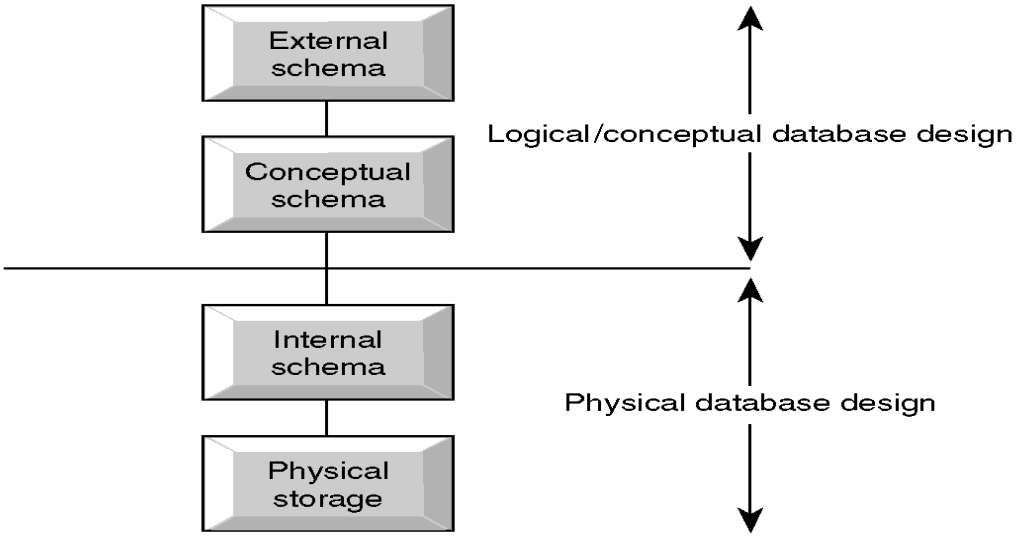
database, like adding tables, adding and deleting an attribute, changing the specification of the objects in the database.

All the above and many other functionalities are possible due to the three level ANSI-SPARC Database System Architectures.



Three-level ANSI-SPARC Architecture of a Database System

ANSI-SPARC Architecture and Database Design Phases



The Database System Architecture is consists of the three levels: External level, conceptual level, Internal level.

External Level:

The external level is the one closest to the users, i.e., it is the one concerned with the way the data is viewed by individual users. An external view is the content of the database as seen by some particular user (i.e., to that user the database is similar to the view he is working/accessing).

Each external view is defined by a means of an external schema, which consists basically of definitions of each of the various external record types in that external view. The external schema is written using the external DDL portion of the user's data sub language.

External level is users' view of the database. Describes that part of database that is relevant to a particular user. Different users have their own customized view of the database independent of other users.

Conceptual Level:

- The conceptual level is found in between the other two. It is a representation of the entire information content of the database including the relations with one another and security and integrity rules, etc.
- It is the view of the data as it really is or by its entirety rather than as users are forced to see it by the constraints of (for example) the particular language or hardware they might be using.
- The conceptual view is defined by means of the conceptual schema, which is written using another DDL, the conceptual DDL of the data sublanguage in use. If data independence is to be achieved, then those conceptual DDL must not involve any considerations of storage structure or access technique. Thus there must be no reference in the conceptual schema to stored field representations, stored record sequence, indexing, hashing addressing, pointers or any other storage and access details.
- The conceptual schema includes a great many additional features, such as the security and integrity rules.

Conceptual level is community view of the database. Describes what data is stored in database and relationships among the data.

Internal Level:

Is the one closest to the physical storage, i.e., it is concerned with the way the data is physically stored.

- Is a low-level representation of the entire database?

The internal view is described by means of the internal schema, which not only defines the various stored record types but also specifies what indexes exist, how stored fields are represented, what physical sequence the stored records are in, and so on. The internal schema is written using yet another DDL-the internal DDL.

There will be many distinct external views, each consisting of a more or less abstract representation of some portion of the total database, and there will be precisely one conceptual view, consisting of a similarly abstract representation of the database in its entirety. Note that most users will not be interested in the total database, but only in some restricted portion of it. Likewise, there will be precisely one internal view, representing the total database as physically stored. The following example will clarify the levels to some extent.

At the conceptual level, the database contains information concerning an entity type called employee. Each individual employee occurrence has an employee_number (six characters), a department_number (four characters), and a salary (five decimal digits).

At the internal level, employees are represented by a stored record type called stored_emp, twenty bytes long. Stored_emp contains four stored fields: a six byte prefix (presumably containing control information such as flags or pointers), and three data fields corresponding to the three properties of employees. In addition, stored_emp records are indexed on the empno field by an index called empx, whose definition is not shown.

The Pascal user has an external view of the database in which employee is represented by a Pascal record containing two fields (department numbers are of no interest to this user and therefore been omitted from the view). The record type is defined according to the syntax and declaration rules in Pascal.

Similarly, the COBOL user has an external view in which each employee is represented by a COBOL record containing two fields (this time salary is not needed by this user and omitted). The record type is defined according to COBOL rules.

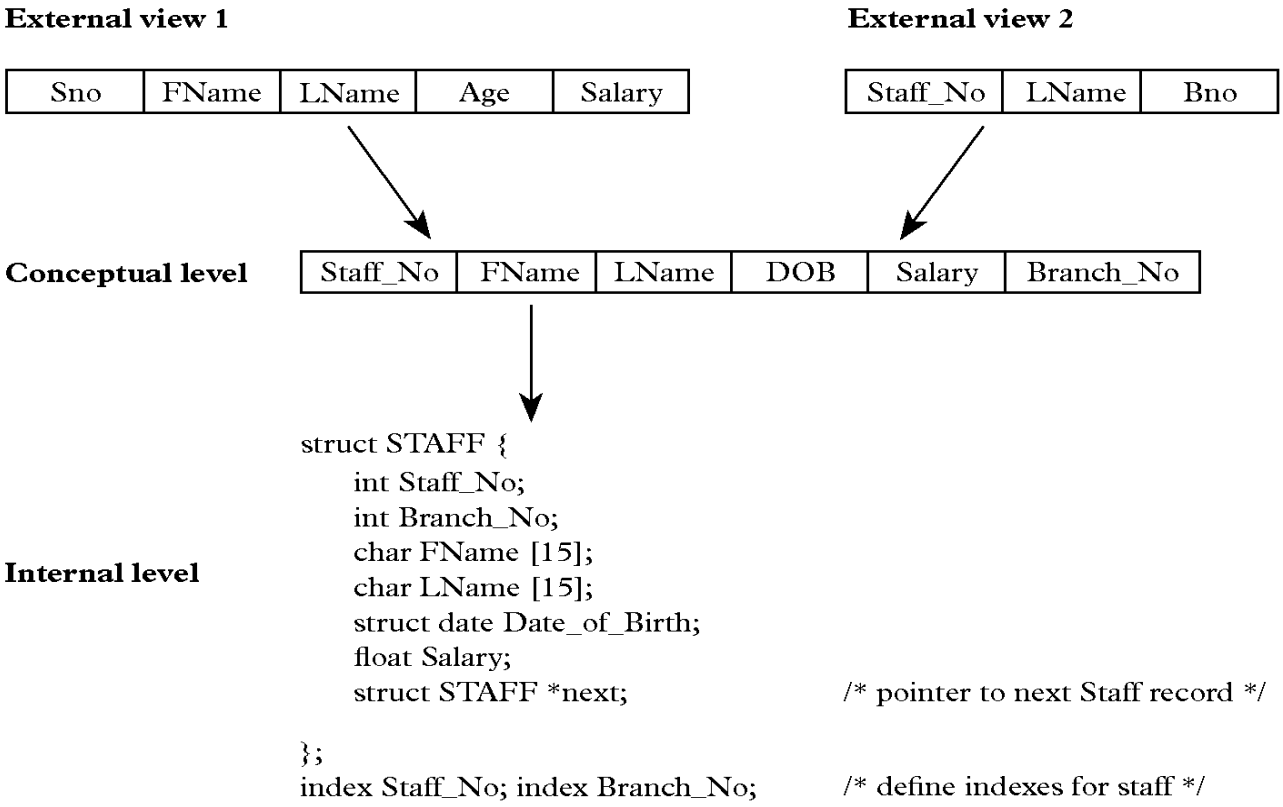
Notice that: the corresponding objects can have different names at each level. The employee number is referred to as empno in the Pascal view, as emp# in the internal view and as employee_number in the conceptual view. In general, to define the correspondence between the conceptual view and the internal view; and the conceptual view and the external view we need an operation called mapping. The mappings are important, for example, fields can have different data types, field and record names can be changed, and several conceptual fields can be combined into a single external field, and so on.

Internal level is the physical representation of the database on the computer. Describes how the data is stored in the database.

The following example can be taken as an illustration for the difference
between the three levels in the ANSI-SPARC database system Architecture. Where:

- The first level is concerned about the group of users and their respective data requirement independent of the other.
- The second level is describing the whole content of the database where one piece of information will be represented once.
- The third level explains the coorsponding physical description of each data requierment.

Differences between Three Levels of ANSI-SPARC Architecture



Defines DBS schemas at three levels:

- **Internal schema:** at the internal level to describe physical storage structures and access paths. Typically uses a physical data model.
- **Conceptual schema:** at the conceptual level to describe the structure and constraints for the whole database for a community of users. Uses a conceptual or an implementation data model.
- **External schema:** at the external level to describe the various user views. Usually uses the same data model as the conceptual level.

Data Independence

Define as the ability (immunity) of applications to change storage structure and access technique without modifying the main application.

In older systems, the way in which the data is organized in secondary storage, and the technique for accessing it, are both dictated by the requirements of the application under consideration, and moreover that knowledge of that data organization and that access technique is built into the application logic and code. In such type of systems it is impossible to change the storage structures (how the data is physically stored) or access technique (how it is accessed) without affecting the application. The applications mentioned are simply programs that are designed to specific tasks where every knowledge of the data structure and the access mechanism is also defined within itself.

In database systems, it would be extremely undesirable to allow applications to be data dependent. Major reasons are:

Different applications will need different views of the same data. Suppose, we have an employee data stored with (employee_id, employee_name, employee_salary, and employee_address, etc. data items), one user may need only to use the employee_name and employee_salary data items whereas another user require only the employee_name and employee_address data items. For data dependent applications, such needs will entail the change of the main application with creation of two different copies of the same application, as it would be applied by both users.

The Database Administrator (DBA) must have the freedom to change the storage structure or access technique in response to changing requirements, without having to modify existing applications. For example, new kinds of data might be added to the database, new standards might be adopted; new types of storage devices might become available, and so on.

Logical Data Independence:

- Refers to immunity of external schemas to changes in conceptual schema.
- Conceptual schema changes e.g. addition or removal of entities should not require changes to external schema or rewrites of application programs.

- The capacity to change the conceptual schema without having to change the external schemas and their application programs.

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.
- The capacity to change the internal schema without having to change the conceptual schema
- Refers to immunity of conceptual schema to changes in the internal schema
- Internal schema changes e.g. using different file organizations, storage structures/devices should not require change to conceptual or external schemas.

Data Independence and the ANSI-SPARC Three-level Architecture