Chapter 1

Introduction to Database System

Database systems are designed to manage large data set in an organization. The data management involves both definition and the manipulation of the data which ranges from simple representation of the data to considerations of structures for the storage of information. The data management also consider the provision of mechanisms for the manipulation of information.

Today, Databases are essential to every business. They are used to maintain internal records, to present data to customers and clients on the World-Wide-Web, and to support many other commercial processes. Databases are likewise found at the core of many modern organizations.

The power of databases comes from a body of knowledge and technology that has developed over several decades and is embodied in specialized software called a database management system, or DBMS. A DBMS is a powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely. These systems are among the most complex types of software available.

Thus, for our question: What is a database? In essence a database is nothing more than a collection of shared information that exists over a long period of time, often many years. In common dialect, the term database refers to a collection of data that is managed by a DBMS.

Thus the DB course is about:

- How to organize data
- Supporting multiple users
- Efficient and effective data retrieval
- Secured and reliable storage of data
- Maintaining consistent data
- Making information useful for decision making

Data management passes through the different levels of development along with the development in technology and services. These levels could best be described by categorizing the levels into three levels of development. Even though there is an advantage and a problem overcome at each new level, all methods of data handling are in use to some extent. The major three levels are;

1. **Manual Approach**
2. **Traditional File Based Approach**
3. **Database Approach**

1. Manual Approach

In the manual approach, data storage and retrieval follows the primitive and traditional way of information handling where cards and paper are used for the purpose. The data storage and retrieval will be performed using human labour.

- Files, for as many event and objects as the organization has, are used to store information.
- Each of the files containing various kinds of information is labelled and stored in one or more cabinets.
- The cabinets could be kept in safe places for security purpose based on the sensitivity of the information contained in it.
- Insertion and retrieval is done by searching first for the right cabinet then for the right the file then the information.
- One could have an indexing system to facilitate access to the data

Limitations of the Manual approach

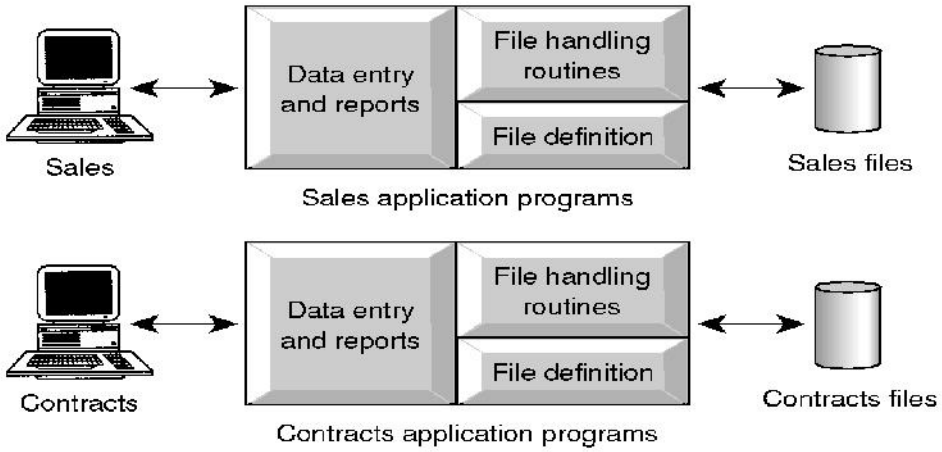
- Prone to error
- Difficult to update, retrieve, integrate
- You have the data but it is difficult to compile the information
- Limited to small size information
- *Cross referencing is difficult*

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 base on where the data resides in the system.

2. Traditional 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. There were, and still are, several computer applications with file based processing used for the purpose of data handling. Even though the approach evolved over time, the basic structure is still similar if not identical.

- File based systems were an early attempt to computerize the manual filing system.
- This approach is the decentralized computerized data handling method.
- A collection of application programs perform 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.
- Since every application defines and manages its own data, the system is subject to serious data duplication problem.
- File, in traditional file based approach, is a collection of records which contains logically related data.



Sales Files

Property_for_Rent(Property Number, Street, Area, City, Post Code, Property Type, Number of Rooms, Monthly Rent, Owner Number)

Owner(Owner Number, First Name, Last Name, Address, Telephone Number)

Renter(Renter Number, First Name, Last Name, Address, Telephone Number, Preferred Type, Maximum Rent)

Contracts Files

Lease(Lease Number, Property Number, Renter Number, Monthly Rent, Payment Method, Deposit, Paid, Rent Start Date, Rent Finish Date, Duration)

Property_for_Rent(Property Number, Street, Area, City, Post Code, Monthly Rent)

Renter(Renter Number, First Name, Last Name, Address, Telephone Number)

Limitations of the Traditional File Based approach

As business application become more complex 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 or Isolation of Data: Available information in one application may not be known. Data Synchronisation is done manually.
- Limited data sharing- every application maintains its own data.
- Lengthy development and maintenance time
- Duplication or redundancy of data (*money and time cost* and *loss of data integrity*)
- Data dependency on the application- data structure is embedded in the application; hence, a change in the data structure needs to change the application as well.
- Incompatible file formats or data structures (e.g. "C" and COBOL) between different applications and programs creating inconsistency and difficulty to process jointly.
- Fixed query processing which is defined during application development

The limitations for the traditional file based data handling approach arise from two basic reasons.

1. Definition of the data is embedded in the application program which makes it difficult to modify the database definition easily.
2. No control over the access and manipulation of the data beyond that imposed by the application programs.

The most significant problem experienced by the traditional file based approach of data handling can be formalized by what is called **"update anomalies"**. We have three types of update anomalies;

1. **Modification Anomalies**: a problem experienced when one ore more data value is modified on one application program but not on others containing the same data set.
2. **Deletion Anomalies**: a problem encountered where one record set is deleted from one application but remain untouched in other application programs.
3. **Insertion Anomalies**: a problem experienced when ever there is new data item to be recorded, and the recording is not made in all the applications. And when same data item is inserted at different applications, there could be errors in encoding which makes the new data item to be considered as a totally different object.

3. Database Approach

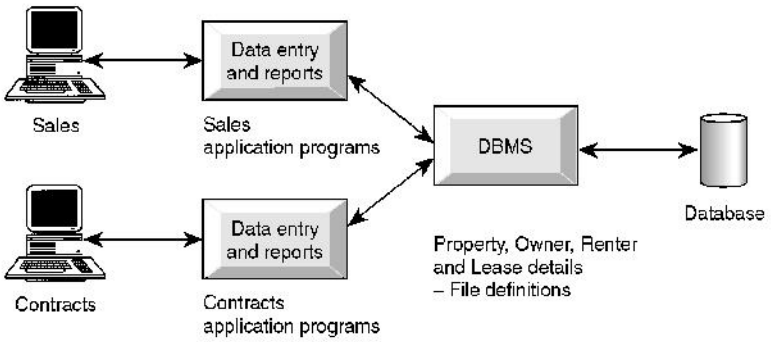
Following a famous paper written by Dr. Edgar Frank Codd in 1970, database systems changed significantly. Codd proposed that database systems should present the user with a view of data organized as tables called relations. Behind the scenes, there might be a complex data structure that allowed rapid response to a variety of queries. But, unlike the user of earlier database systems, the user of a relational system would not be concerned with the storage structure. Queries could be expressed in a very high-level language, which greatly increased the efficiency of database programmers. The database approach emphasizes the integration and sharing of data throughout the 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 and description of 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 is known as "**Metadata**" or "**Data Dictionary**" or "**Systems Catalogue**" or "**Data about Data**" or some times "**Data Directory**".
- Since a database contains information about the data (metadata), it is called a self *descriptive collection* of 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 deigned 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.

Benefits of the database approach

- *Data can be shared*: two or more users can access and use same data instead of storing data in redundant manner for each user.
- *Improved accessibility of data*: by using structured query languages, the users can easily access data without programming experience.
- *Redundancy can be reduced*: isolated data is integrated in database to decrease the redundant data stored at different applications.
- *Quality data can be maintained*: the different integrity constraints in the database approach will maintain the quality leading to better decision making
- *Inconsistency can be avoided*: controlled data redundancy will avoid inconsistency of the data in the database to some extent.
- *Transaction support can be provided*: basic demands of any transaction support systems are implanted in a full scale DBMS.
- *Integrity can be maintained*: data at different applications will be integrated together with additional constraints to facilitate *validity* and *consistency* of shared data resource.
- *Security measures can be enforced*: the shared data can be secured by having different levels of clearance and other data security mechanisms.
- *Improved decision support*: the database will provide information useful for decision making.
- *Standards can be enforced*: the different ways of using and dealing with data by different unite of an organization can be balanced and standardized by using database approach.
- *Compactness*: since it is an electronic data handling method, the data is stored compactly (no voluminous papers).
- *Speed*: data storage and retrieval is fast as it will be using the modern fast computer systems.
- *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.



Property_for_Rent(Property Number, Street, Area, City, Post Code, Property Type, Number of Rooms, Monthly Rent, Owner Number)

Owner(Owner Number, First Name, Last Name, Address, Telephone Number)

Renter(Renter Number, First Name, Last Name, Address, Telephone Number), Preferred Type, Maximum Rent)

Lease(Lease Number, Property Number, Renter Number, Payment Method, Deposit, Paid, Rent Start Date, Rent Finish Date)

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.

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
10. ***Utility services***: sets of utility service facilities like
    - Importing data
    - Statistical analysis support
    - Index reorganization
    - Garbage collection

DBMS and Components of DBMS Environment

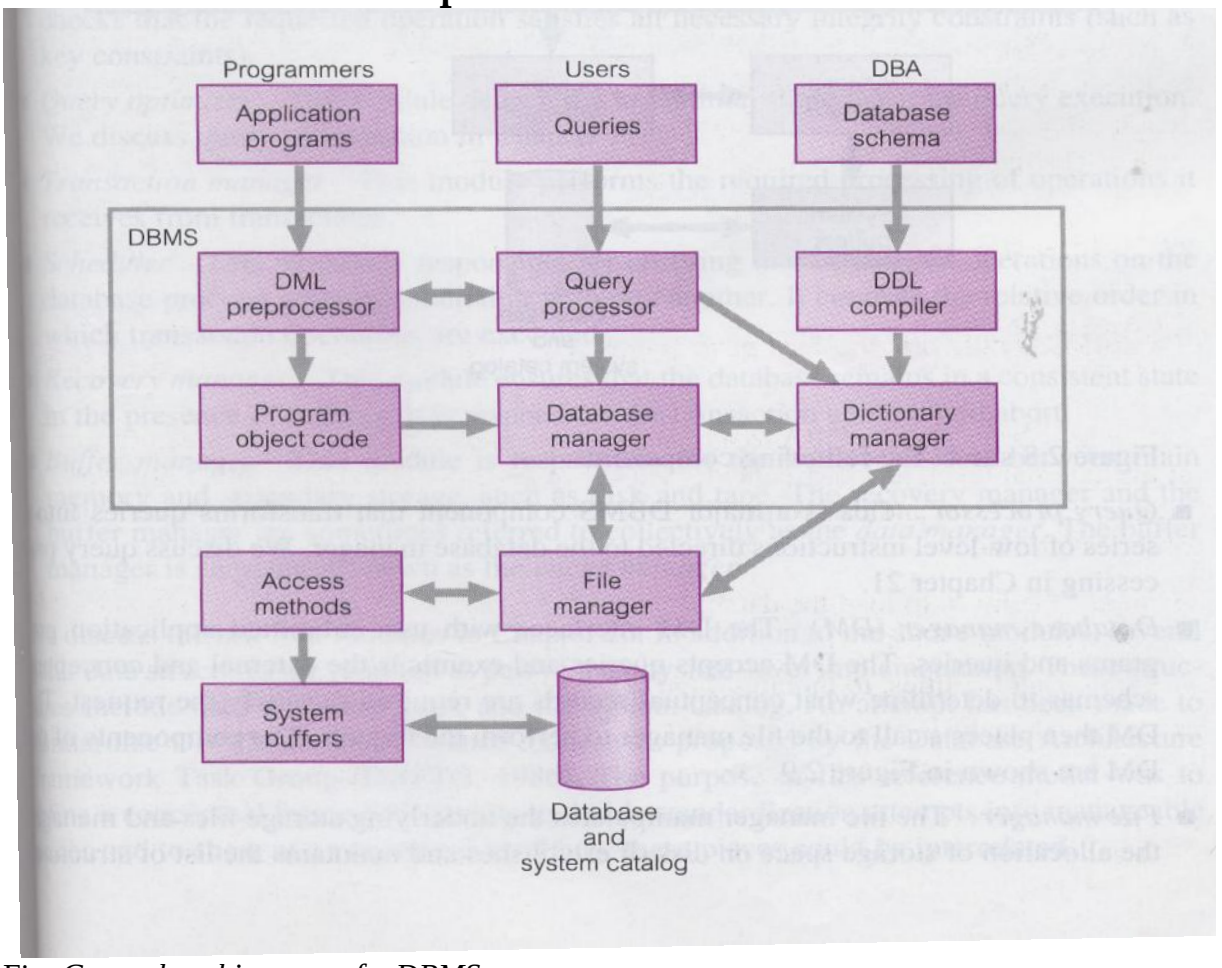


Fig. *General architecture of a DBMS*

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

- **Data Manipulation Language (DML):**
  - Is a core command used by end-users and programmers to store, retrieve, and access the data in the database e.g. SQL
  - Since the required data or Query by the user will be extracted using this type of language, it is also called "Query Language"

- **Data Dictionary:**
  - 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:**
  - 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

The DBMS is software package that helps to design, manage, and use data using the database approach. Taking a DBMS as a system, one can describe it with respect to it environment or other systems interacting with the DBMS. The DBMS environment has five components. To design and use a database, there will be the interaction or integration of Hardware, Software, Data, Procedure and People.

1. **Hardware**: are components that one can touch and feel. These components are comprised of various types of personal computers, mainframe or any server computers to be used in multi-user system, network infrastructure, and other peripherals required in the system.

2. **Software**: are collection of commands and programs used to manipulate the hardware to perform a function. These include components like the DBMS software, application programs, operating systems, network software, language software and other relevant software.

3. **Data**: since the goal of any database system is to have better control of the data and making data useful, Data is the most important component to the user of the database. There are two categories of data in any database

system: that is *Operational* and *Metadata*. Operational data is the data actually stored in the system to be used by the user. Metadata is the data that is used to store information about the database itself.
The structure of the data in the database is called the ***schema***, which is composed of the *Entities*, *Properties of entities*, and *relationship between entities and business constraints*.

4. **Procedure**: this is the rules and regulations on *how to design and use* a database. It includes procedures like how to log on to the DBMS, how to use facilities, how to start and stop DBMS, how to make backup, how to treat hardware and software failure, how to change the structure of the database.

5. **People**: this component is composed of the people in the organization that are responsible or play a role in designing, implementing, managing, administering and using the resources in the database. This component includes group of people with high level of knowledge about the database and the design technology to other with no knowledge of the system except using the data in the database.

Database Development Life Cycle (DDLC)

As it is one component in most information system development tasks, there are several steps in designing a database system. Here more emphasis is given to the design phases of the system development life cycle. The major steps in database design 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 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 data model* to implement the data structure.
        - It is particular DBMS **independent** and with no other physical considerations.
    c. **Physical Design**: physical implementation of the logical 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. Tuning the database operations for best performance.

Roles in Database Design and Use

As people are one of the components in DBMS 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.

   1. **Data Administrator (DA)**: is responsible on management of data resources. This involves in database planning, development, maintenance of standards policies and procedures at the conceptual and logical design phases.

   2. **Database Administrator (DBA)**: This is more technically oriented role. DBA is responsible for the physical realization of the database. It is involved in physical design, implementation, security and integrity control of the database.

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.

1. **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.
2. **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.
   - The application programmer 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.

4. **End Users**

   Workers, whose job requires accessing the database frequently for various purposes, there are different group of users in this category.

   1. **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.
   2. **Sophisticated Users**
      - 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

3. **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.

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

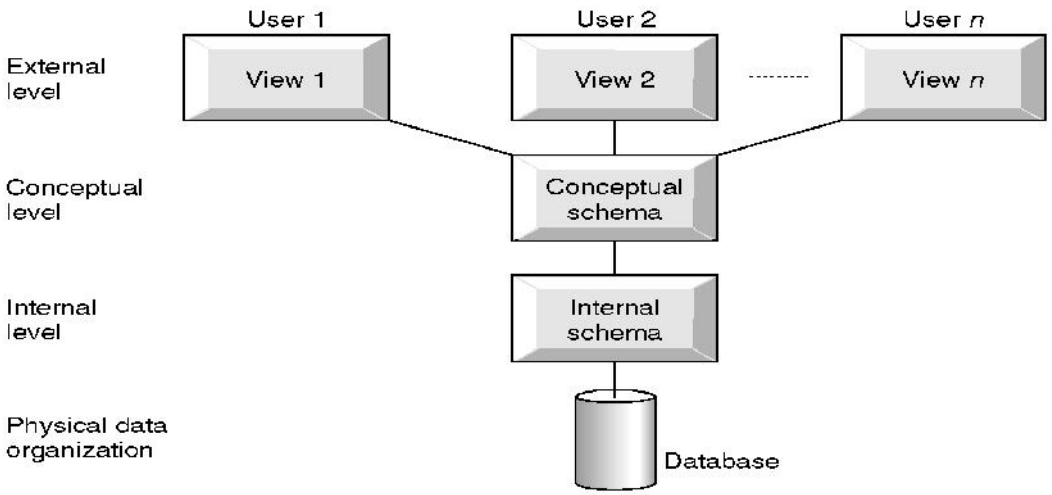
- **DBMS** designers and implementers: who design and implement different DBMS software.
- **Tool Developers**: experts who develop software packages that facilitates database system designing and use. Prototype, simulation, code generator developers could be an example. Independent software vendors 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.

ANSI-SPARC Architecture

The purpose and origin of the Three-Level database architecture

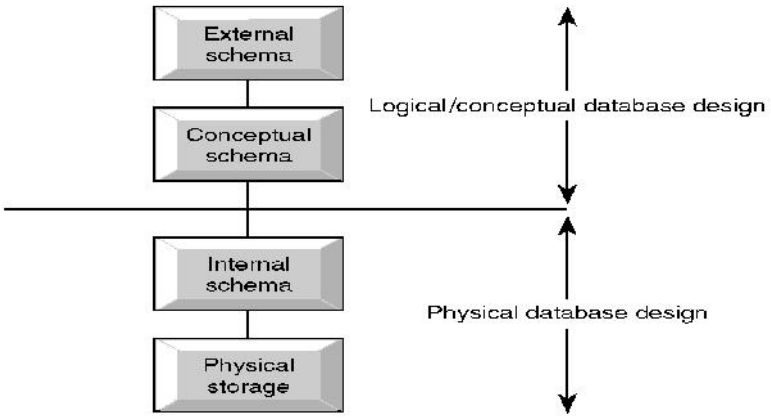
- All users should be able to access same data. This is important since the database is having a shared data feature where all the data is stored in one location and all users will have their own customized way of interacting with the data.
- A user's view is unaffected or immune to changes made in other 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, such as change of hard disk
- 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 of the above and much more functionalities are possible due to the three level ANSI-SPARC architecture.



Three-level ANSI-SPARC Architecture of a Database

ANSI-SPARC Architecture and Database Design Phases



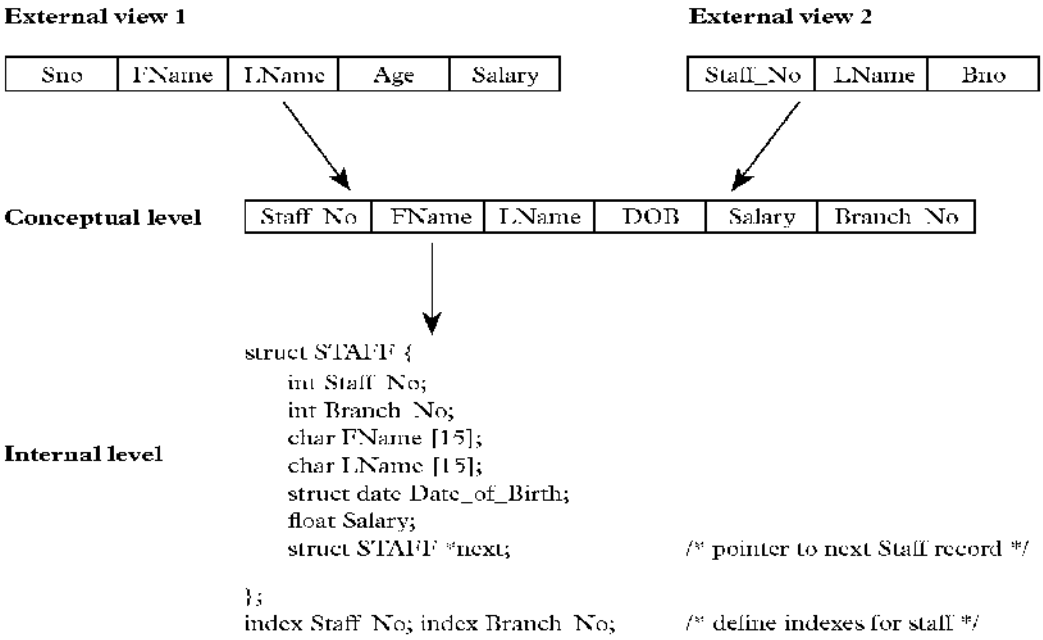
***External Level*:** Users' view of the database. It describes that part of database that is relevant to a particular user. Different users have their own customized view of the database independent of other users.

***Conceptual Level*:** Community view of the database. Describes what data is stored in database and relationships among the data along with the business constraints.

***Internal Level*:** 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 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



Differences between Three Levels of ANSI-SPARC Architecture

Data Independence

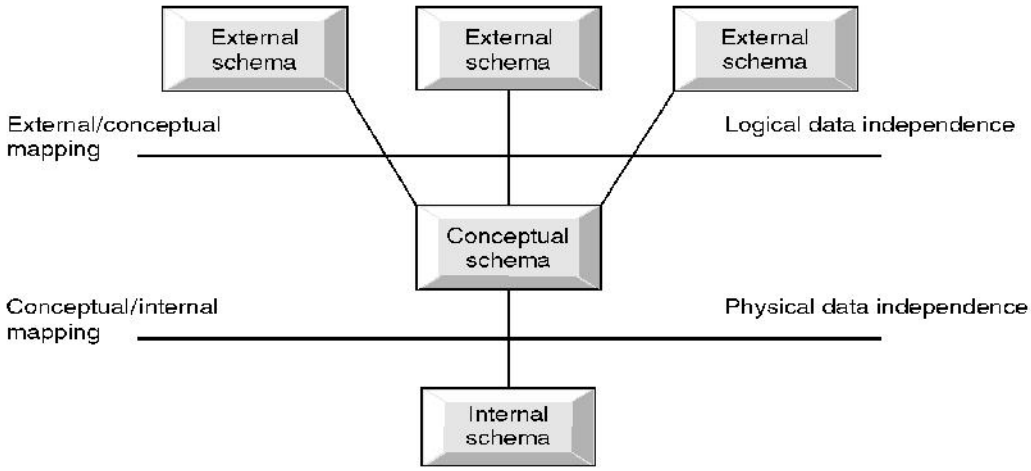
Logical Data Independence:

- Refers to immunity of external schemas to changes in conceptual schema.
- Conceptual schema changes e.g. addition/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

Database Languages

Data Definition Language (DDL)

- 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)

- Provides basic data manipulation operations on data held in the database.
- Language for accessing and manipulating the data organized by the appropriate data model
- DML also known as query language

Procedural DML: user specifies what data is required and how to get the data.

Non-Procedural DML: user specifies what data is required but not how it is to be retrieved

Data Control Language (DCL)

- Allows a DBA to define access control and privileges for users.
- It is a mechanism for implementing security at a database object level.
- Uses the Grant and Revoke SQL Statements

SQL is the most widely used non-procedural query language

Fourth Generation Language (4GL)

- Query Languages
- Forms Generators
- Report Generators
- Graphics Generators
- Application Generators

A Classification of data models

Data Model

A specific DBMS has its own specific Data Definition Language to define a database schema, but this type of language is too low level to describe the data requirements of an organization in a way that is readily understandable by a variety of users.

We need a higher-level language. Such a higher-level description of the database schema is called *data-model*.

Data Model: a set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.

A **data model** is a description of the way that data is stored in a database. Data model helps to understand the relationship between entities and to create the most effective structure to hold data.

Data Model is a collection of tools or concepts for describing

- Data
- Data relationships
- Data semantics
- Data constraints

The main ***purpose*** of Data Model is to represent the data in an understandable way.

Categories of data models include:

- Object-based
- Record-based
- Physical

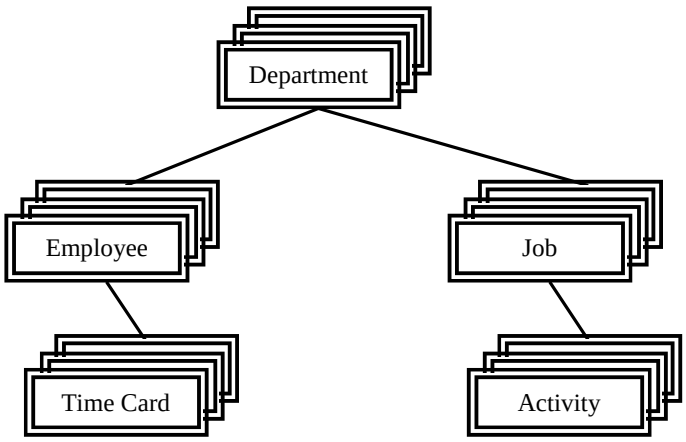
Record-based Data Models

Consist of a number of fixed format records.
Each record type defines a fixed number of fields,
Each field is typically of a fixed length.

- Hierarchical Data Model
- Network Data Model
- Relational Data Model

1. Hierarchical Model

- The simplest data model
- Record type is referred to as node or segment
- The top node is the root node
- Nodes are arranged in a hierarchical structure as sort of upside-down tree
- A parent node can have more than one child node
- A child node can only have one parent node
- The relationship between parent and child is one-to-many
- Relation is established by creating physical link between stored records (each is stored with a predefined access path to other records)
- To add new record type or relationship, the database must be redefined and then stored in a new form.



ADVANTAGES of Hierarchical Data Model:

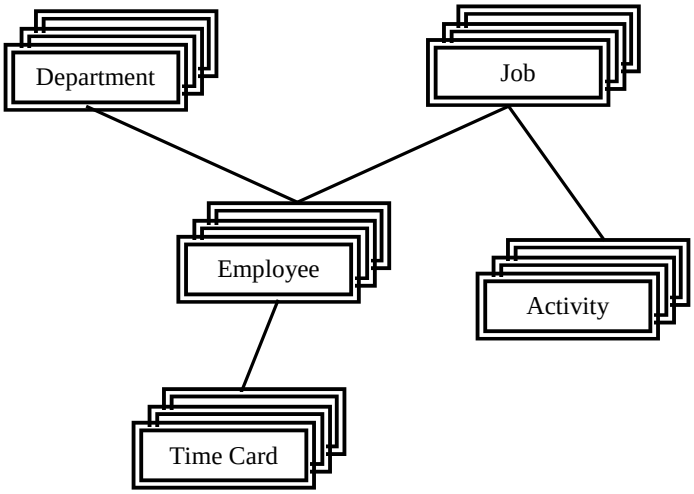
- Hierarchical Model is simple to construct and operate on
- Corresponds to a number of natural hierarchically organized domains - e.g., assemblies in manufacturing, personnel organization in companies
- Language is simple; uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT etc.

DISADVANTAGES of Hierarchical Data Model:

- Navigational and procedural nature of processing
- Database is visualized as a linear arrangement of records
- Little scope for "query optimization"

2. Network Model

- Allows record types to have more than one parent unlike hierarchical model
- A network data models sees records as set members
- Each set has an owner and one or more members
- Allow no many to many relationship between entities
- Like hierarchical model network model is a collection of physically linked records.
- Allow member records to have more than one owner



ADVANTAGES of Network Data Model:

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

DISADVANTAGES of Network Data Model:

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

3. Relational Data Model

- Developed by **Dr. Edgar Frank Codd in 1970** (famous paper, 'A Relational Model for Large Shared Data Banks')
- Terminologies originates from the branch of mathematics called set theory and predicate logic and is based on the mathematical concept called ***Relation***
- Can define more flexible and complex relationship
- Viewed as a collection of tables called "Relations" equivalent to collection of record types
- **Relation**: Two dimensional table
- Stores information or data in the form of tables → **rows and columns**
- A **row** of the table is called tuple→ equivalent to **record**
- A **column** of a table is called attribute→ equivalent to **fields**
- Data value is the value of the Attribute
- Records are related by the data stored jointly in the fields of records in two tables or files. The related tables contain information that creates the relation
- The tables seem to be independent but are related some how.
- No physical consideration of the storage is required by the user
- Many tables are merged together to come up with a new virtual view of the relationship

Alternative terminologies		
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

- The rows represent records (collections of information about separate items)
- The columns represent fields (particular attributes of a record)
- Conducts searches by using data in specified columns of one table to find additional data in another table
- In conducting searches, a relational database matches information from a field in one table with information in a corresponding field of another table to produce a third table that combines requested data from both tables