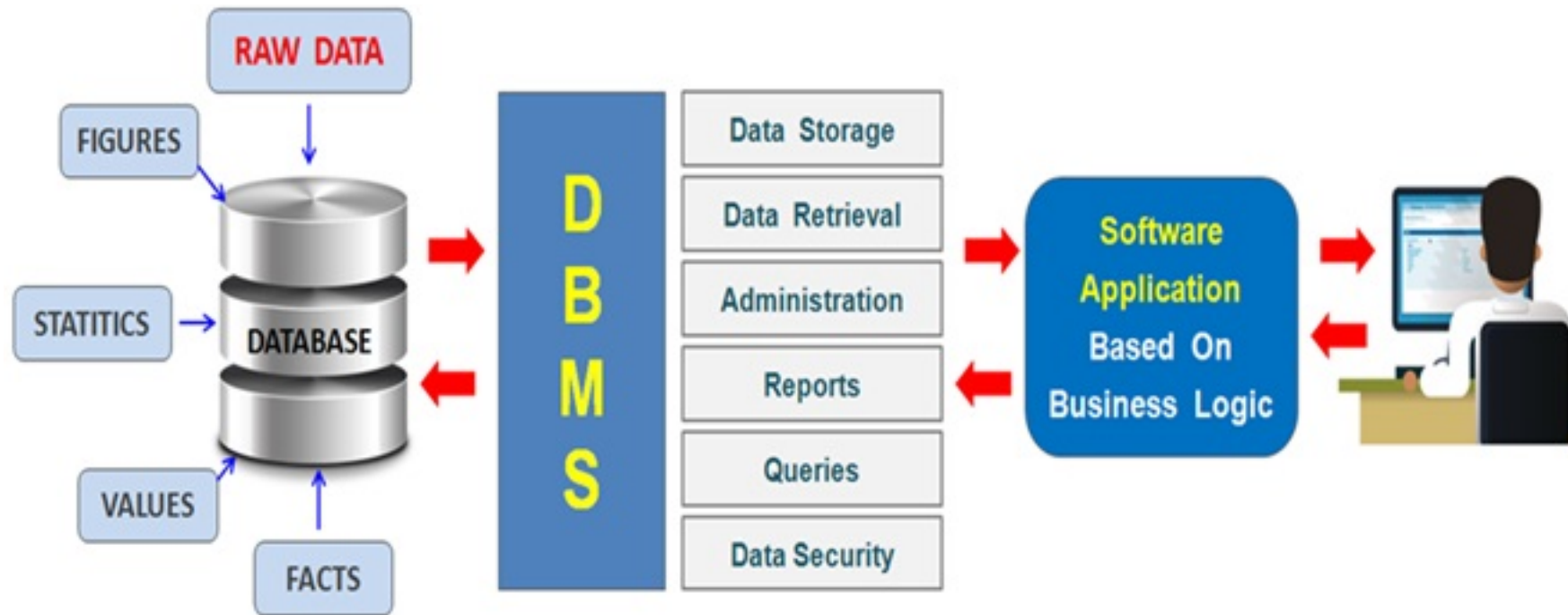


LS-01. DBMS Introduction



- What is a database, and what is a DBMS?
- What are the differences between DBMS and Filesystem?
- Why should we consider a DBMS to manage data?
- What is a data model, and what is a semantical (conceptual, logical) data model?

1. Database Management System (DBMS) Definition

Problem

- The volume of information available to us is exploding, and the cost of such data is recognized as an asset (active).
- We need tools that simplify the tasks of managing data and extracting useful information in an efficient fashion.
- Otherwise, the cost of acquiring and managing data far exceeds the value derived from it.

Database is a **collection of data**, typically describing activities of one or more **related** organizations.

Most of today's websites and platforms utilize databases:

- Amazon: to hold information about the consumers and the products.
- Netflix: to hold information about the registered users and current movies.
- **University database** (as an example) might contain information about:

Entities: students, faculty, courses, and classrooms

Attributes of entities: student name, course number, and classroom location

Relationships between entities: students' enrollment in course, faculty teaching courses, and use classrooms for courses

Database Management System (DBMS) is a **software designed to assist** users and developers maintaining and utilizing large collection of data.

- This course explains the basics of DBMS such as its architecture, data models, data schemas, data independence, E-R model, relation model, relational database design, and storage and file structure and much more.

MySQL, Oracle and Microsoft SQL server are examples of DBMS.

- In this course, we will use **MySQL** as DBMS to create and maintain databases.

Different types of DBMS are in use, but we – in this course – focus on **relational database systems** that are the dominant type of DBMS today.

The alternative to using a DBMS is to store the data in files (**file systems**) and develop application-specific code to manage it.

1.1. DBMS vs Filesystem

Scenario:

- **Big company** with a large collection (say 500 GB) of data on employees, departments, products and sales.
- This collection of data is accessed **concurrently** by several employees.
- Questions about the data must be answered **quickly**, changes made to the data by different users must be applied **consistently**, and access to certain parts of the data (e.g., salaries) must be **restricted**.

1.1.1. First solution: File System - Bad

We can try to manage the data by storing it in the operating system's files, however:

- We have to write special programs to answer **each question a user may want to ask** about the data.
- We must manually protect the data from **inconsistent changes** made by different users accessing the data concurrently.
- Data **redundancy** and multiple file **formats** increase complexity of accessing data.
- We must ensure the data is restored to a consistent state **if the systems crashes**.
- The operating system provides only password mechanisms, that is not sufficient to **enforce security policies** where different users have permissions to access different subsets of the data in one file.

1.1.2. Second solution: DBMS - Good

- By storing data in a DBMS, we can utilize a set of features (detailed in the next slide) to manage data in a robust and efficient manner.
- As the volume of data and the number of users grow, DBMS support becomes indispensable.

1.2. DBMS Characteristics

Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks.

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management.

A **database management system** stores data in such a way that it becomes easier to retrieve, manipulate, and produce information. A modern DBMS has the following characteristics:

- 1.2.1. **Real-world entity** – A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.
- 1.2.2. **Relation-based tables** – DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
- 1.2.3. **Isolation of data and application** – A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.
- 1.2.4. **Less redundancy** – DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
- 1.2.5. **Consistency** – Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.
- 1.2.6. **Query Language** – DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
- 1.2.7. **ACID Properties** – DBMS follows the concepts of **A**tomicity, **C**onsistency, **I**solation, and **D**urability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.

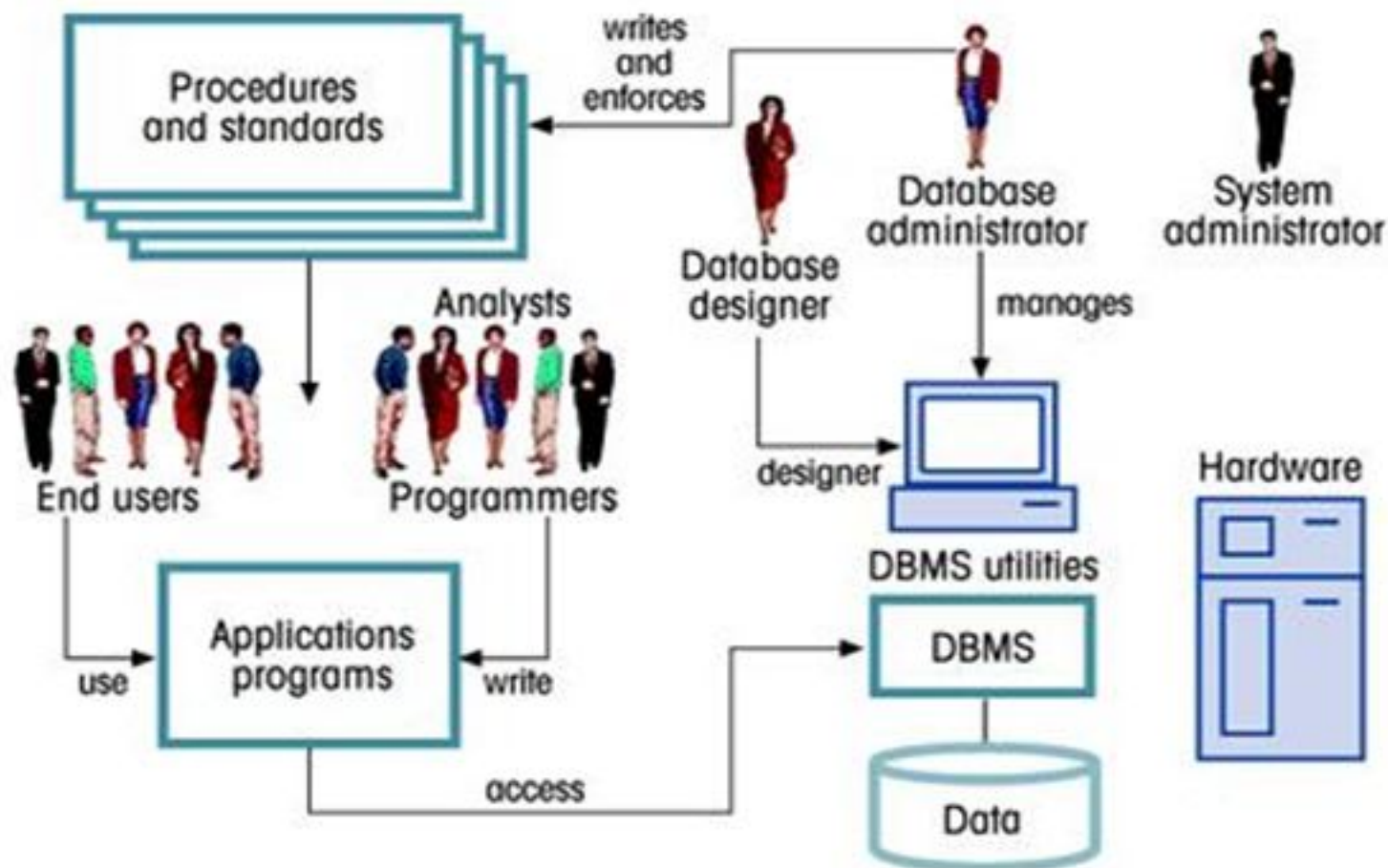
- 1.2.8. **Multiuser and Concurrent Access** – DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.
- 1.2.9. **Multiple views** – DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.
- 1.2.10. **Security** – Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

Is there a reason *not to use* a DBMS? - Yes!

- DBMS is a complex piece of software that is optimized for certain kinds of workloads (e.g., answering complex queries, handling many concurrent requests).
- Such complex software may not be adequate for ***certain specialized applications*** with a very tight real-time constraints or with a few well-defined operations where ***an efficient custom code*** must be used instead.

1.3. DBMS Users

A typical DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows



The Database System Environment

1.3.1. Administrators – Database Administrator (DBA) maintain the DBMS and are responsible for administering the database. They are responsible to look after its usage and by whom it should be used. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.

DBA - a person who has central control over the system, whose functions are:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users

1.3.2. Designers – Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in what format. They identify and design the whole set of entities, relations, constraints, and views.

- **Application programmers** -- are computer professionals who write application programs.
- **Specialized users** --write specialized database applications that do not fit into the traditional data-processing framework. For example, CAD, graphic data, audio, video.

1.3.3. End Users – End users are those who actually reap the benefits of having a DBMS. End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

- **Naive users** -- unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
- **Sophisticated users** -- interact with the system without writing programs
 - using a database query language
 - or by using tools such as data analysis software.

2. DBMS – Architecture & Abstraction Schemes

2.1. DBMS Design

The design of a DBMS depends on its architecture; design can be:

- **Centralized,**
- **Decentralized hierarchical,**
- **Distributed.**

2.2. DBMS Tier Architectures

The architecture of an system can be seen as either single tier or multi-tier (levels). An n-tier architecture divides the whole system into related but independent **n** modules (levels), which can be independently modified, altered, changed, or replaced.

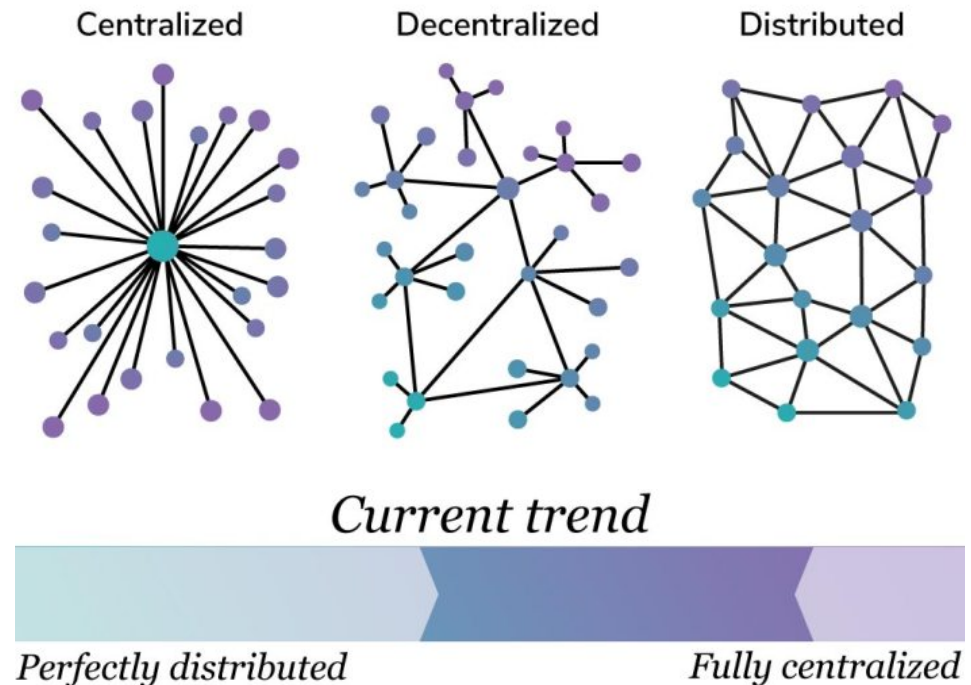
1-tier DBMS Architecture

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

2-tier DBMS Architecture

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

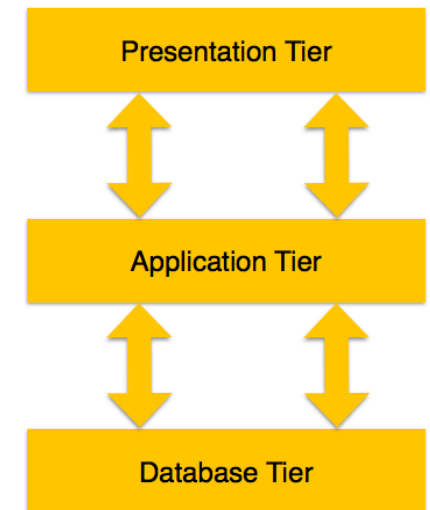
3-tier DBMS Architecture



A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

- **User (Presentation) Tier** – End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.
- **Application (Middle) Tier** – At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **Database (Data) Tier** – At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

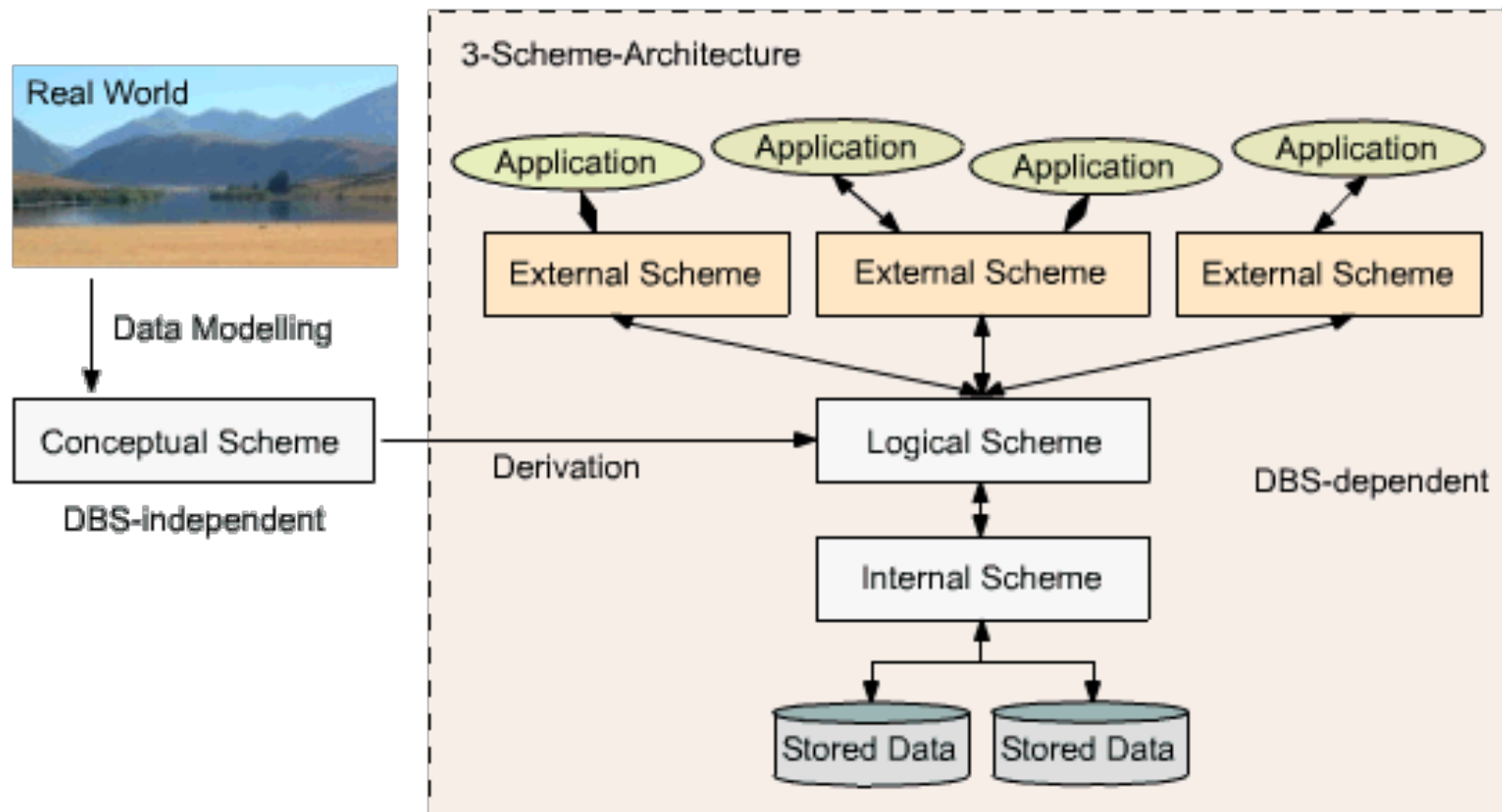


2.3. Four Abstraction Schemes in a DBMS

2.3.1. Conceptual Scheme is independent of DBS and is as close as possible to the real world.

2.3.2. Logical Scheme (derived from the conceptual scheme) describes the basic construction of the data structure. This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

2.3.3. Internal Scheme. The internal data scheme describes the content of the data and the required service functionality which is used for the operation of the DBMS. Internal scheme describes the physical grouping of the data and the use of the storage space, therefore, the internal scheme describes the data from a view very close to the computer or system in general. It completes the logical scheme with data technical aspects like storage methods or help functions for more efficiency.

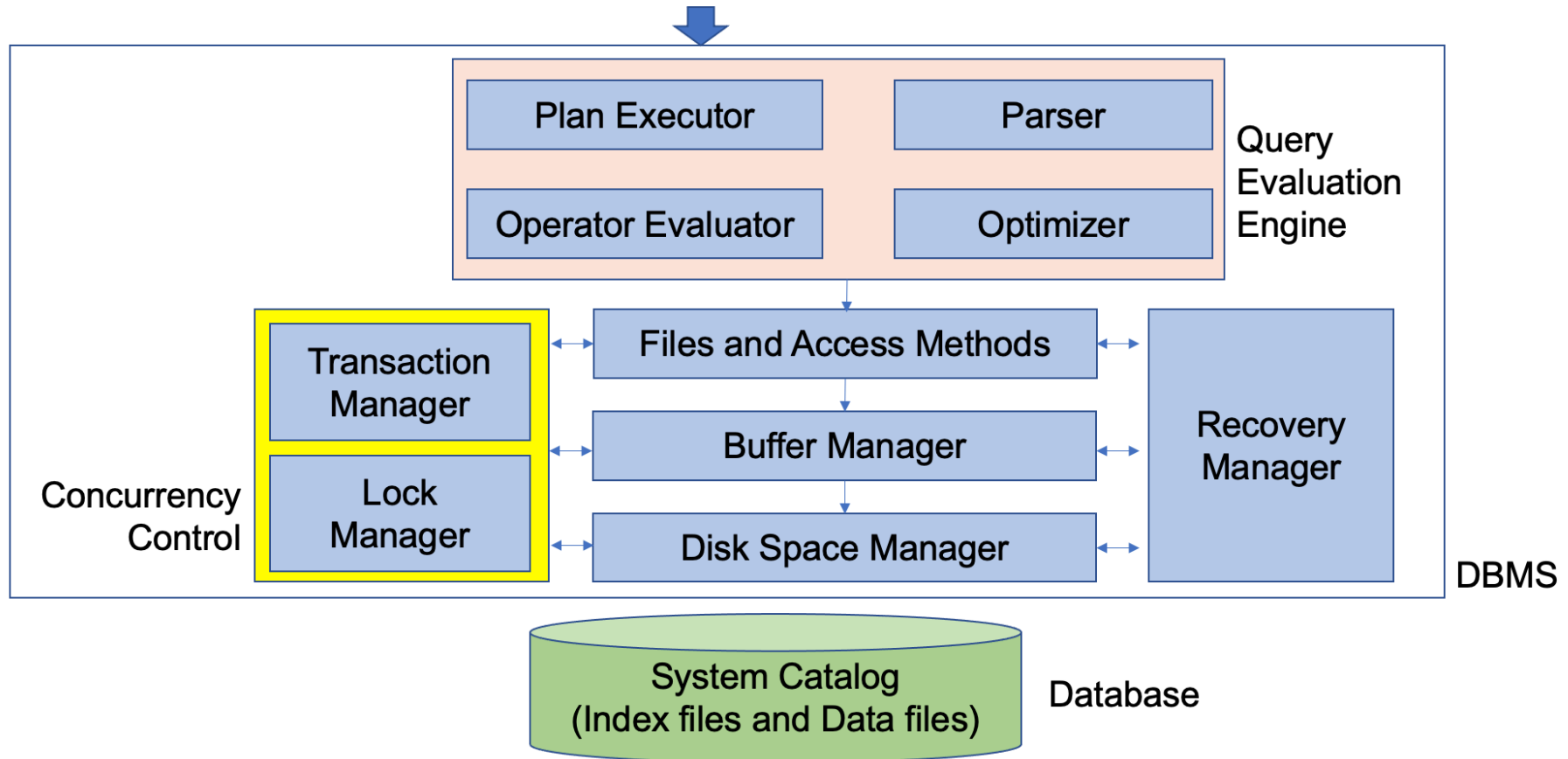


2.3.4. External Scheme. An external data scheme describes the information about the user view of specific users and the specific methods and constraints connected with this information. The external scheme of a specific application, generally, only highlights that part of the logical scheme which is relevant for its application. Therefore, a database has exactly one internal and one logical scheme but may have several external schemes for several applications using this database.

The aim of the three-schemes architecture is the separation of the user applications from the physical database, the stored data. Physically the data is only existent on the internal level while other forms of representation are calculated or derived respectively if needed. The DBMS has the task to realize this representation between each of these levels.

2.4. Functional Architecture of a DBMS

SQL Commands from Web forms, application front ends, SQL interfaces



3. DBMS – Semantic Data Models

Data models define how the logical structure of a DB is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

What is a semantic data model?

- A semantic data model is a more abstract model that makes it easier for a user to come up with a **good initial description** of the data in an enterprise.
- Defining the semantical data model is the start point in **designing a database**, that is subsequently **translated into a data model** that DBMS actually supports and implements.
- A widely used semantic data model called the **entity-relationship (ER) model** allows us to define the different entities and draw relationships among them.
- Earlier data models were not so scientific, hence they were prone to introduce lots of **duplication** and **anomalies**.

Country Name	Continent	Population	Area
Montenegro	Europe	630548	14026
Afghanistan	Asia	29863000	647500
Albania	Europe	3581655	28748
Algeria	Africa	32854000	2381740
Andorra	Europe	67313	468
Angola	Africa	15941000	1246700
Antigua and Barbuda	North America	69108	443
Azerbaijan	Europe	8327618	86600
Argentina	South America	38747000	2791810
Australia	Australia and Oceania	20555300	7686850
Austria	Europe	8189000	83871
Bahamas	North America	303770	13940
Bahrain	Asia	727000	665

3.1. Flat File Data Models

The very first data model could be flat data-models, where all the data used are to be kept in the same plane in single table.

The flat (or table) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another.

Flat File Model represented using a **Data Dictionary**.

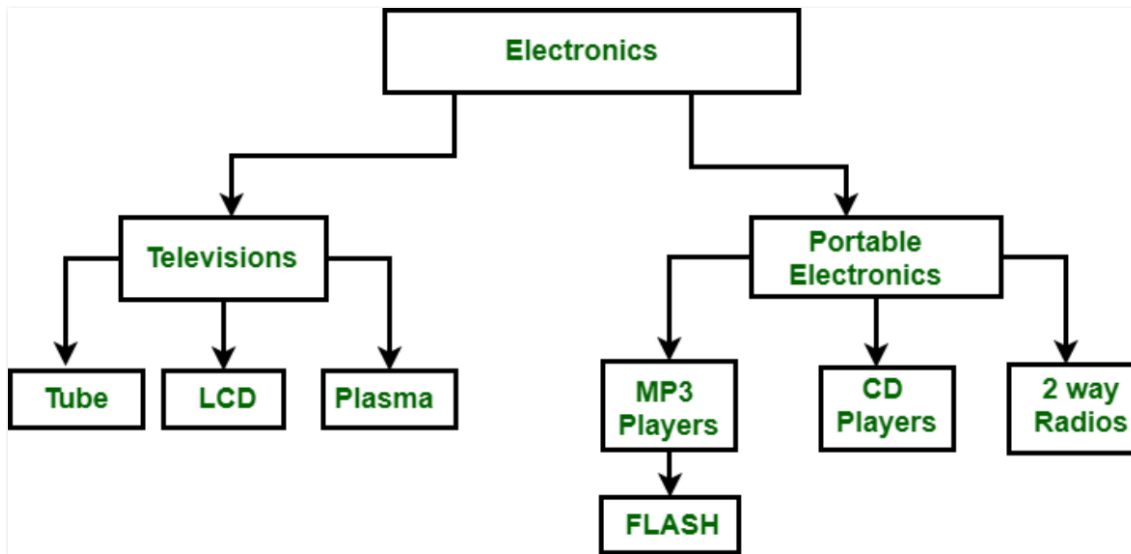
Example of **Flat File Set** and **Data Dictionary**.

Field Name	Data Type	Data Format	Field Size	Description	Example
License ID	Integer	NNNNNN	6	Unique number ID for all drivers	12345
Surname	Text		20	Surname for Driver	Jones
First Name	Text		20	First Name for Driver	Arnold
Address	Text		50	First Name for Driver	11 Rocky st Como 2233
Phone No.	Text		10	License holders contact number	0400111222
D.O.B	Date / Time	DD/MM/YYYY	10	Drivers Date of Birth	08/05/1956

3.2. Hierarchical Model

Hierarchical data model is the oldest type of the data model. It was developed by IBM in 1968. It organizes data in the tree-like structure. Hierarchical model consists of the the following :

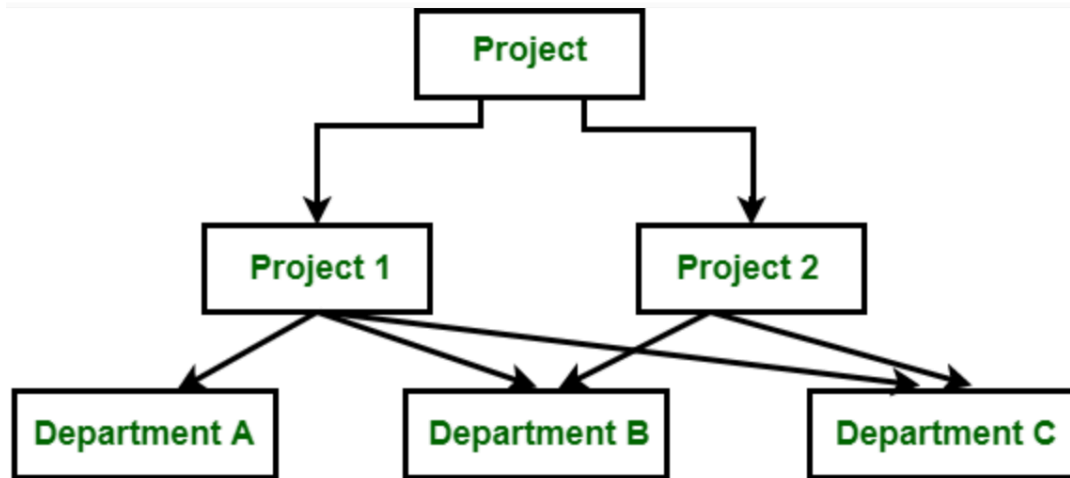
- It contains nodes which are connected by branches.
- The topmost node is called the root node.
- If there are multiple nodes appear at the top level, then these can be called as root segments.
- Each node has exactly one parent.
- One parent may have many child.



In the above figure, Electronics is the root node which has two children i.e. Televisions and Portable Electronics. These two has further children for which they act as parent. For example: Television has children as Tube, LCD and Plasma, for these three Television act as parent. It follows one to many relationship.

3.3. Network Model

It is the advance version of the hierarchical data model. To organize data it uses directed graphs instead of the tree-structure. In this child can have more than one parent. It uses the concept of the two data structures i.e. Records and Sets.



In the above figure, Project is the root node which has two children i.e. Project 1 and Project 2. Project 1 has 3 children and Project 2 has 2 children. Total there are 5 children i.e. Department A, Department B and Department C, they are network related children as we said that this model can have more than one parent. So, for the Department B and Department C have two parents i.e. Project 1 and Project 2.

3.4. Other Models

- Object oriented Model – Describe data at the conceptual and view levels.
- Record based logical Models - describe data at the conceptual and view levels. These models specify logical structure of database with records, fields and attributes.
- A document-oriented database, or document store, is a computer program and data storage system designed for storing, retrieving and managing document-oriented information, also known as semi-structured data.

3.5. ER Model

Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

Conceptual ER Model is based on:

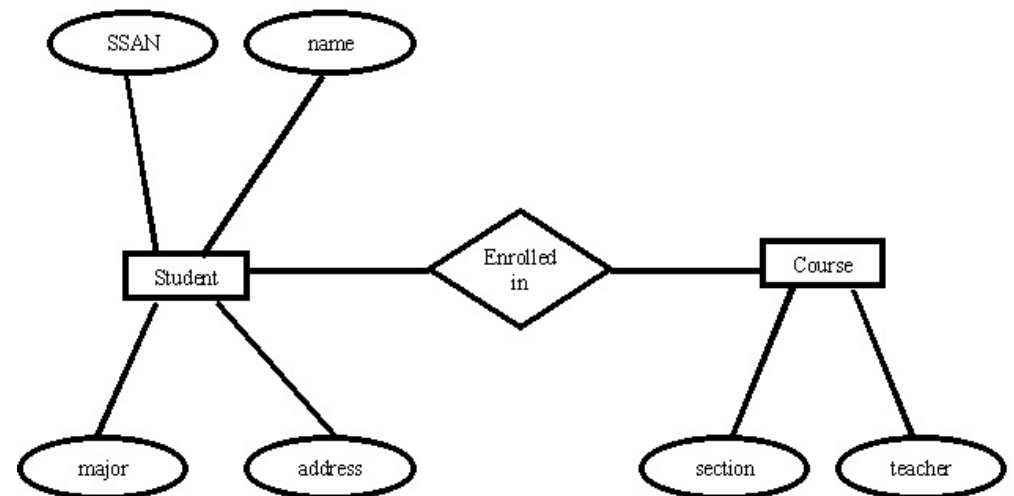
- **Entity** – An entity in an ER Model is a real-world entity
- **Attribute** – Entity having properties called attributes.
- **Domain** – Every attribute is defined by its set of values called domain.
- **Relationship** – The logical association among entities is called *relationship*. Relationships are mapped with entities in various ways.

Mapping cardinalities define the number of association between two entities:

- one to one
- one to many
- many to one
- many to many

ER Model represented as ER Diagram

For example, in a university database, a student is considered as an entity. Student has various attributes like name, age, address, group, etc.

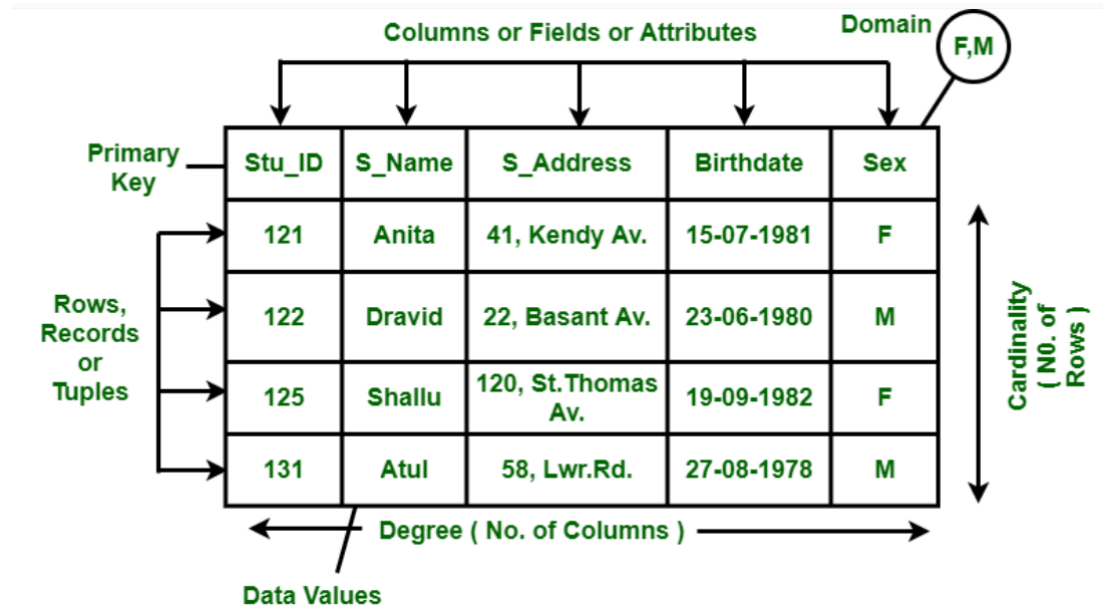


3.6. Relational Model

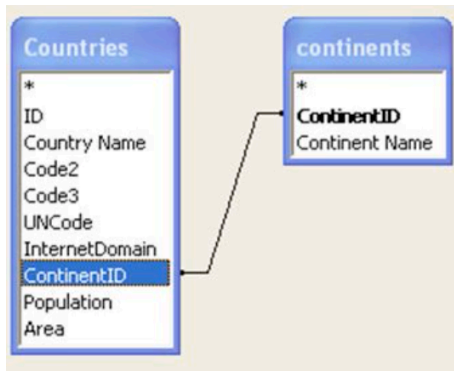
The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on **first-order predicate logic** and defines a table as an **n-ary relation**.

The main highlights of this model are

- Data is stored in tables with specific characteristics called **relations**.
- Relations can be **normalized**.
- In normalized relations, values saved are **atomic** values in cells.
- In normalized relations many **anomaly** has removed.
- Each **tuple** (row, record) in a relation contains a **unique** value.
- Each **attribute set** (column, field) in a relation contains values from a same **domain** (and datatype).



Relation Model represented as Relation Diagram or as DB Schema



3.7. Database Schema

A **database schema** is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.



A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

The schema for a relation specifies:

- The name of the relation.
- The name of each **field** (or attribute or column).
- The **type of data** that can be stored of each field.

The student information in a university database may be stored in a relation with the **following schema**:

Student (sid: string, name: string, age: integer, gpa: real)

 Relation Name  Attribute: Data Type GPA - grade point average

- This schema says that each record in the student relation has four fields, with field names and types as indicated.
- Think of the schema as a **template** for describing a student, and each row in the Students relation is a **record** that describes a student in the university.

3.8. Database Instance

It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

- An example **instance** of the Students relation can be:

sid	name	age	gpa
5324	Jones	18	3.4
5388	Smith	18	3.2
5360	Madayan	19	3.8

3.9. Integrity Constraints

- We can make the description of the relation to be **more precise** by specifying some constraints.
- Integrity constraints are **conditions** that any record in this relation **must satisfy**.
- For example: every student in the university has a unique *sid* value.
- The expressiveness of these constraints is an **important aspect of the data model**.