Module-I

What is Database

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

For example: The college Database organizes the data about the admin, staff, students and faculty etc.

Using the database, you can easily retrieve, insert, and delete the information.

Database Management System

- Database management system is a software which is used to manage the database. For example: MySQL, Oracle, etc are a very popular commercial database which is used in different applications.
- DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
- It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

DBMS allows users the following tasks:

- **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
- **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.
- **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
- **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

Characteristics of DBMS

- It uses a digital repository established on a server to store and manage the information.
- It can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contains ACID properties which maintain data in a healthy state in case of failure.

- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirements of the user.

Key Features of DBMS

- **Data modeling:** A DBMS provides tools for creating and modifying data models, which define the structure and relationships of the data in a database.
- **Data storage and retrieval:** A DBMS is responsible for storing and retrieving data from the database, and can provide various methods for searching and querying the data.
- **Concurrency control:** A DBMS provides mechanisms for controlling concurrent access to the database, to ensure that multiple users can access the data without conflicting with each other.
- **Data integrity and security:** A DBMS provides tools for enforcing data integrity and security constraints, such as constraints on the values of data and access controls that restrict who can access the data.
- **Backup and recovery:** A DBMS provides mechanisms for backing up and recovering the data in the event of a system failure.
- **DBMS can be classified into two types:** Relational Database Management System (RDBMS) and Non-Relational Database Management System (NoSQL or Non-SQL)
- **RDBMS:** Data is organized in the form of tables and each table has a set of rows and columns. The data are related to each other through primary and foreign keys.
- **NoSQL:** Data is organized in the form of key-value pairs, documents, graphs, or column-based. These are designed to handle large-scale, high-performance scenarios.

A database is a collection of interrelated data which helps in the efficient retrieval, insertion, and deletion of data from the database and organizes the data in the form of tables, views, schemas, reports, etc. For Example, a university database organizes the data about students, faculty, admin staff, etc. which helps in the efficient retrieval, insertion, and deletion of data from it.

Database Languages

- Data Definition Language
- Data Manipulation Language
- Data Control Language
- Transactional Control Language

Data Definition Language

DDL is the short name for Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

- **CREATE:** to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- **ALTER:** alters the structure of the existing database
- **DROP:** delete objects from the database

- **TRUNCATE:** remove all records from a table, including all spaces allocated for the records are removed
- **COMMENT:** add comments to the data dictionary
- **RENAME:** rename an object

Data Manipulation Language

DML is the short name for Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database. **Data query language(DQL)** is the subset of "Data Manipulation Language". The most common command of DQL is **SELECT** statement. SELECT statement help on retrieving the data from the table without changing anything in the table.

- **SELECT:** retrieve data from a database
- **INSERT:** insert data into a table
- **UPDATE:** updates existing data within a table
- **DELETE:** Delete all records from a database table
- **MERGE:** UPSERT operation (insert or update)
- **CALL:** call a PL/SQL or Java subprogram
- **EXPLAIN PLAN:** interpretation of the data access path
- **LOCK TABLE:** concurrency Control

Data Control Language

DCL is short for Data Control Language which acts as an access specifier to the database.(basically to grant and revoke permissions to users in the database

- **GRANT:** grant permissions to the user for running DML(SELECT, INSERT, DELETE,…) commands on the table
- **REVOKE:** revoke permissions to the user for running DML(SELECT, INSERT, DELETE,…) command on the specified table

Transactional Control Language

TCL is short for Transactional Control Language which acts as an manager for all types of transactional data and all transactions. Some of the command of TCL are

- **Roll Back:** Used to cancel or Undo changes made in the database
- **Commit:** It is used to apply or save changes in the database
- **Save Point:** It is used to save the data on the temporary basis in the database

Data Query Language (DQL):

Data query language(DQL) is the subset of **"Data Manipulation Language"**. The most common command of DQL is 1the **SELECT statement**. SELECT statement helps us in retrieving the data from the table without changing anything or modifying the table. DQL is very important for retrieval of essential data from a database.

Database Management System

The software which is used to manage databases is called Database Management System (DBMS). For Example, MySQL, Oracle, etc. are popular commercial DBMS used in different applications. DBMS allows users the following tasks:

- **Data Definition:** It helps in the creation, modification, and removal of definitions that define the organization of data in the database.
- **Data Updation:** It helps in the insertion, modification, and deletion of the actual data in the database.

- **Data Retrieval:** It helps in the retrieval of data from the database which can be used by applications for various purposes.
- **User Administration:** It helps in registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information corrupted by unexpected failure.

Applications of DBMS:

- **Enterprise Information:** Sales, accounting, human resources, Manufacturing, online retailers.
- **Banking and Finance Sector:** Banks maintaining the customer details, accounts, loans, banking transactions, credit card transactions. Finance: Storing the information about sales and holdings, purchasing of financial stocks and bonds.
- **University:** Maintaining the information about student course enrolled information, student grades, staff roles.
- **Airlines:** Reservations and schedules.
- **Telecommunications:** Prepaid, postpaid bills maintance.

Paradigm Shift from File System to DBMS

File System manages data using files on a hard disk. Users are allowed to create, delete, and update the files according to their requirements. Let us consider the example of file-based University Management System. Data of students is available to their respective Departments, Academics Section, Result Section, Accounts Section, Hostel Office, etc. Some of the data is common for all sections like Roll No, Name, Father Name, Address, and Phone number of students but some data is available to a particular section only like Hostel allotment number which is a part of the hostel office. Let us discuss the issues with this system:

- **Redundancy of data:** Data is said to be redundant if the same data is copied at many places. If a student wants to change their Phone number, he or she has to get it updated in various sections. Similarly, old records must be deleted from all sections representing that student.
- **Inconsistency of Data:** Data is said to be inconsistent if multiple copies of the same data do not match each other. If the Phone number is different in Accounts Section and Academics Section, it will be inconsistent. Inconsistency may be because of typing errors or not updating all copies of the same data.
- **Difficult Data Access:** A user should know the exact location of the file to access data, so the process is very cumbersome and tedious. If the user wants to search the student hostel allotment number of a student from 10000 unsorted students' records, how difficult it can be.
- **Unauthorized Access:** File Systems may lead to unauthorized access to data. If a student gets access to a file having his marks, he can change it in an unauthorized way.
- **No Concurrent Access:** The access of the same data by multiple users at the same time is known as concurrency. The file system does not allow concurrency as data can be accessed by only one user at a time.
- **No Backup and Recovery:** The file system does not incorporate any backup and recovery of data if a file is lost or corrupted.

Advantages of DBMS

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.

- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
- **Size:** It occupies a large space of disks and large memory to run them efficiently.
- **Complexity:** Database system creates additional complexity and requirements.
- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

Data Dictionary

A Data Dictionary comprises two words i.e. **Data** which simply means information being collected through some sources and **Dictionary** means where this information is available. A **Data Dictionary** can be defined as a collection of information on all data elements or contents of databases such as data types, and text descriptions of the system. It makes it easier for users and analysts to use data as well as understand and have common knowledge about inputs, outputs, components of a database, and intermediate calculations.

Metadata

Metadata is simply defined as data about data. It means it is a description and context of the data. It helps to organize, find and understand data. Let me explain to you by giving a real-world example of metadata:

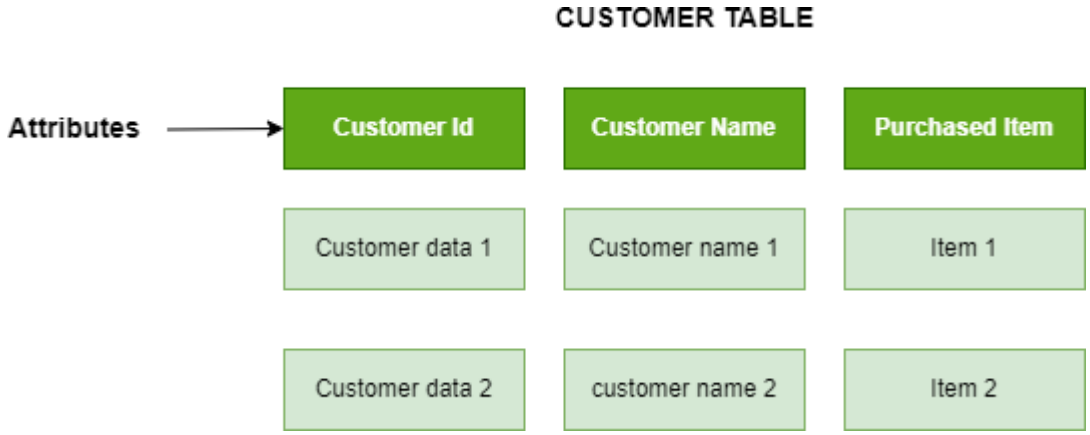
Every time you take a photo with today's cameras a bunch of metadata is gathered and saved with it. Such as

- File name,
- Size of the file,
- Date and time,
- Camera settings etc.

Database Schemas

- The Skeleton of the database is created by the attributes and this skeleton is named Schema.
- Schema mentions the logical constraints like table, primary key, etc.
- The schema does not represent the data type of the attributes.

Details of a Customer



Schema of Customer

Database Schema

- A database schema is a **logical representation of data** that shows how the data in a database should be stored logically. It shows how the data is organized and the relationship between the tables.
- Database schema contains table, field, views and relation between different keys like primary key, foreign key.
- Data are stored in the form of files which is unstructured in nature which makes accessing the data difficult. Thus to resolve the issue the data are organized in structured way with the help of database schema.
- Database schema provides the organization of data and the relationship between the stored data.
- Database schema defines a set of guidelines that control the database along with that it provides information about the way of accessing and modifying the data.

There are 3 types of database schema:

Physical Database Schema

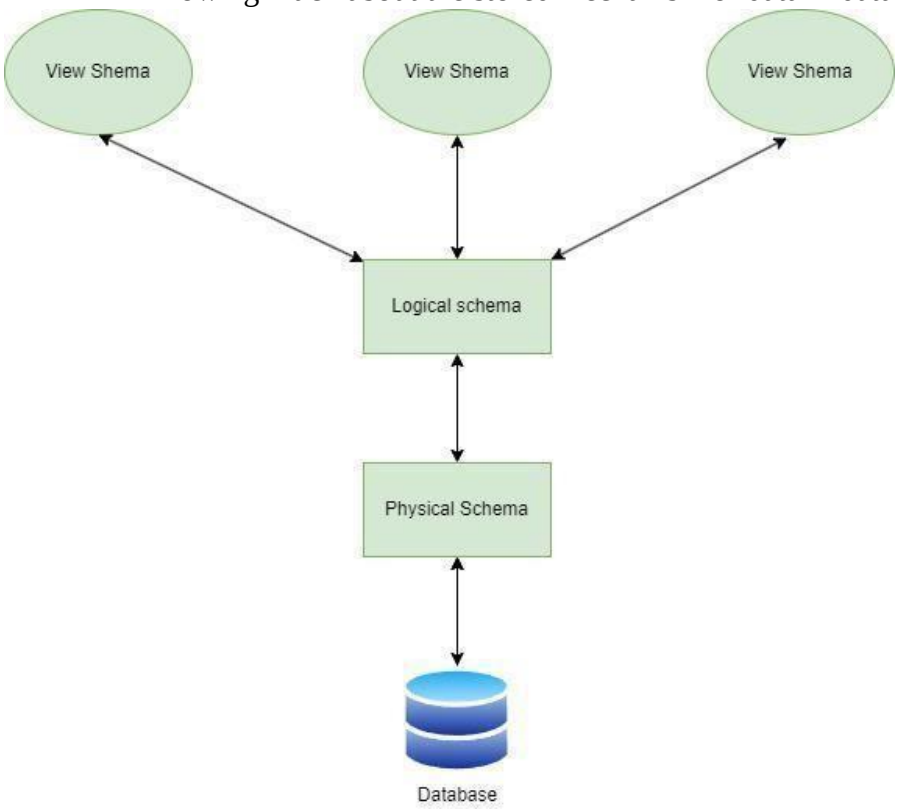
- A Physical schema defines, how the data or information is stored physically in the storage systems in the form of files & indices. This is the actual code or syntax needed to create the structure of a database, we can say that when we design a database at a physical level, it's called physical schema.
- The Database administrator chooses where and how to store the data in the different blocks of storage.

Logical Database Schema

- A logical database schema defines all the logical constraints that need to be applied to the stored data, and also describes tables, views, entity relationships, and integrity constraints.
- The Logical schema describes how the data is stored in the form of tables & how the attributes of a table are connected.
- Using **ER modelling** the relationship between the components of the data is maintained.
- In logical schema different integrity constraints are defined in order to maintain the quality of insertion and update the data.

View Database Schema

- It is a view level design which is able to define the interaction between end-user and database.
- User is able to interact with the database with the help of the interface without knowing much about the stored mechanism of data in database.



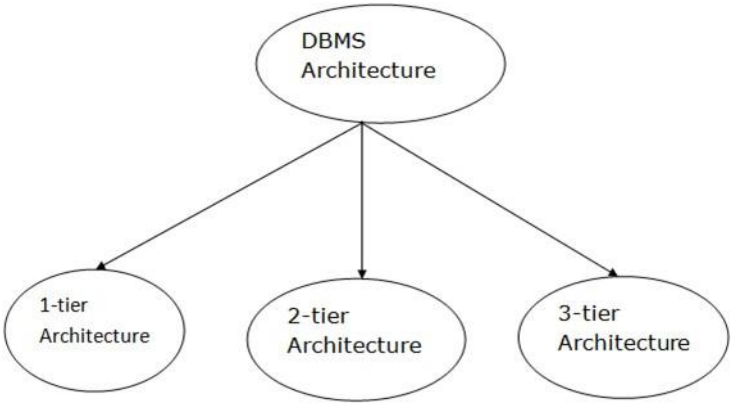
Three Layer Schema Design

DBMS Architecture

- The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network.

- DBMS architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Architecture



Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.

1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

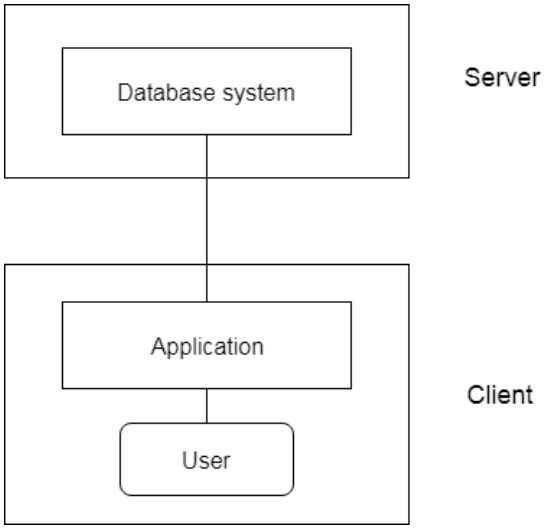


Fig: 2-tier Architecture

3-Tier Architecture

- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

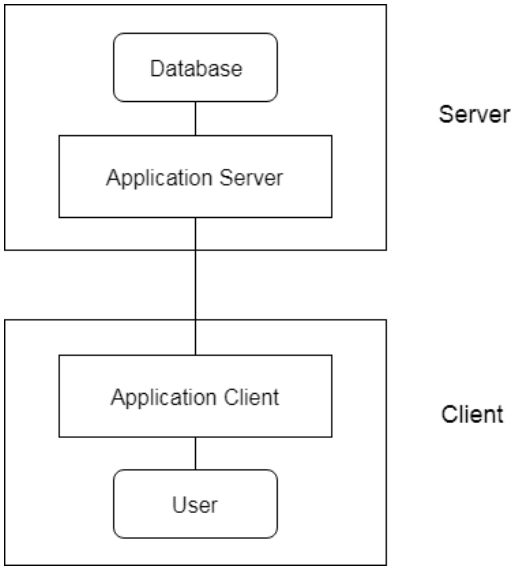
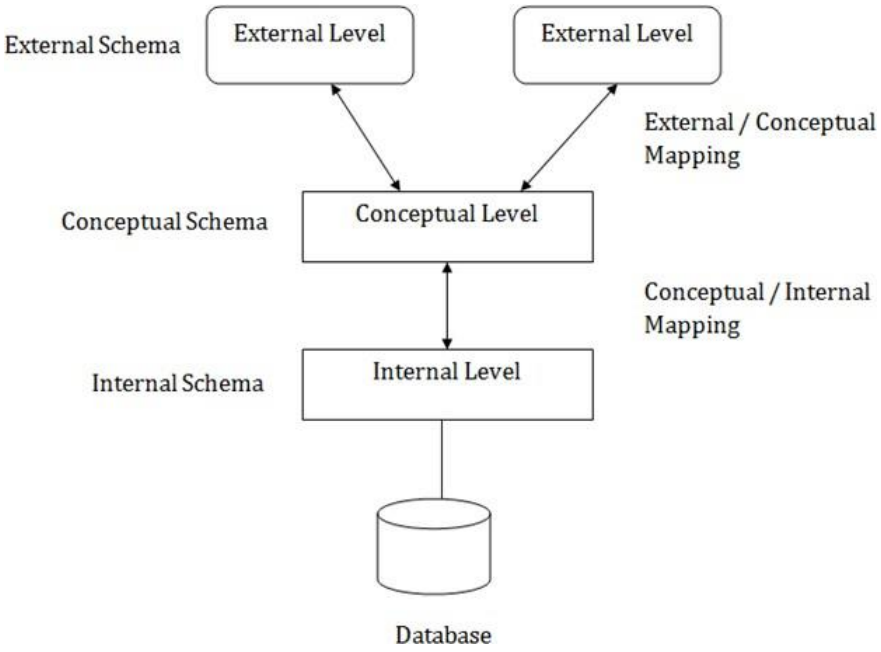


Fig: 3-tier Architecture

Three schema Architecture

- The three schema architecture is also called ANSI/SPARC architecture or three-level architecture.
- This framework is used to describe the structure of a specific database system.
- The three schema architecture is also used to separate the user applications and physical database.
- The three schema architecture contains three-levels. It breaks the database down into three different categories.

The three-schema architecture is as follows:



In the above diagram:

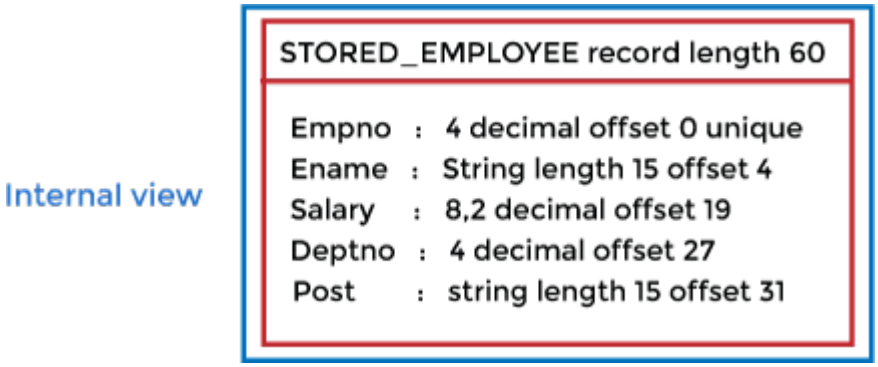
- It shows the DBMS architecture.
- Mapping is used to transform the request and response between various database levels of architecture.
- Mapping is not good for small DBMS because it takes more time.
- In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

Objectives of Three schema Architecture

The main objective of three level architecture is to enable multiple users to access the same data with a personalized view while storing the underlying data only once. Thus it separates the user's view from the physical structure of the database. This separation is desirable for the following reasons:

- Different users need different views of the same data.
- The approach in which a particular user needs to see the data may change over time.
- The users of the database should not worry about the physical implementation and internal workings of the database such as data compression and encryption techniques, hashing, optimization of the internal structures etc.
- All users should be able to access the same data according to their requirements.
- DBA should be able to change the conceptual structure of the database without affecting the user's
- Internal structure of the database should be unaffected by changes to physical aspects of the storage.

1. Internal Level



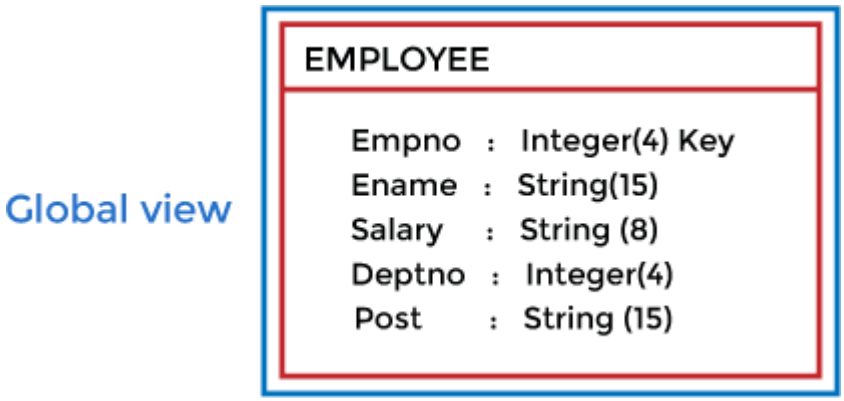
- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.
- It uses the physical data model. It is used to define that how the data will be stored in a block.
- The physical level is used to describe complex low-level data structures in detail.

The internal level is generally is concerned with the following activities:

- Storage space allocations.
  For Example: B-Trees, Hashing etc.

- Access paths.
  For Example: Specification of primary and secondary keys, indexes, pointers and sequencing.
- Data compression and encryption techniques.
- Optimization of internal structures.
- Representation of stored fields.

2. Conceptual Level



- The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- The conceptual schema describes the structure of the whole database.
- The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level.

3. External Level

- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- An external schema is also known as view schema.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems.

Data Abstraction and Data Independence

Database systems comprise complex data structures. In order to make the system efficient in terms of retrieval of data, and reduce complexity in terms of usability of users, developers use abstraction i.e. hide irrelevant details from the users. This approach simplifies database design.

Level of Abstraction in a DBMS

There are mainly 3 levels of data abstraction:

- Physical or Internal Level
- Logical or Conceptual Level
- View or External Level

Physical or Internal Level

This is the lowest level of data abstraction. It tells us how the data is actually stored in memory. Access methods like sequential or random access and file organization methods like B+ trees and hashing are used for the same. Usability, size of memory, and the number of times the records are factors that we need to know while designing the database. Suppose we need to store the details of an employee. Blocks of storage and the amount of memory used for these purposes are kept hidden from the user.

Logical or Conceptual Level

This level comprises the information that is actually stored in the database in the form of tables. It also stores the relationship among the data entities in relatively simple structures. At this level, the information available to the user at the view level is unknown. We can store the various attributes of an employee and relationships, e.g. with the manager can also be stored.

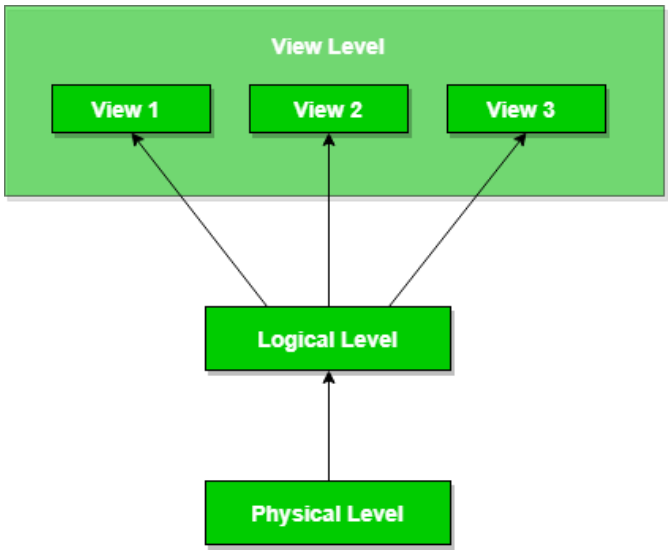
The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as physical data independence. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

View or External Level

This is the highest level of abstraction. Only a part of the actual database is viewed by the users. This level exists to ease the accessibility of the database by an individual user. Users view data in the form of rows and columns. Tables and relations are used to store data. Multiple views of the same database may exist. Users can just view the data and interact with the database, storage and implementation details are hidden from them. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system

Example: In case of storing customer data,

- **Physical level** – it will contains block of storages (bytes,GB,TB,etc)
- **Logical level –** it will contain the fields and the attributes of data.
- **View level** – it works with CLI or GUI access of database



Data Abstraction

The main purpose of data abstraction is to achieve data independence in order to save the time and cost required when the database is modified or altered.

Data Independence

Data Independence is mainly defined as a property of DBMS that helps you to change the database schema at one level of a system without requiring to change the schema at the next level. it helps to keep the data separated from all program that makes use of it. We have namely two levels of data independence arising from these levels of abstraction:

- Physical level data independence
- Logical level data independence

Physical Level Data Independence

It refers to the characteristic of being able to modify the physical schema without any alterations to the conceptual or logical schema, done for optimization purposes, e.g., the Conceptual structure of the database would not be affected by any change in storage size of the database system server. Changing from sequential to random access files is one such example. These alterations or modifications to the physical structure may include:

- Utilizing new storage devices.
- Modifying data structures used for storage.
- Altering indexes or using alternative file organization techniques etc.

Logical Level Data Independence

It refers characteristic of being able to modify the logical schema without affecting the external schema or application program. The user view of the data would not be affected by any changes to the conceptual view of the data. These changes may include insertion or deletion of attributes, altering table structures entities or relationships to the logical schema, etc.

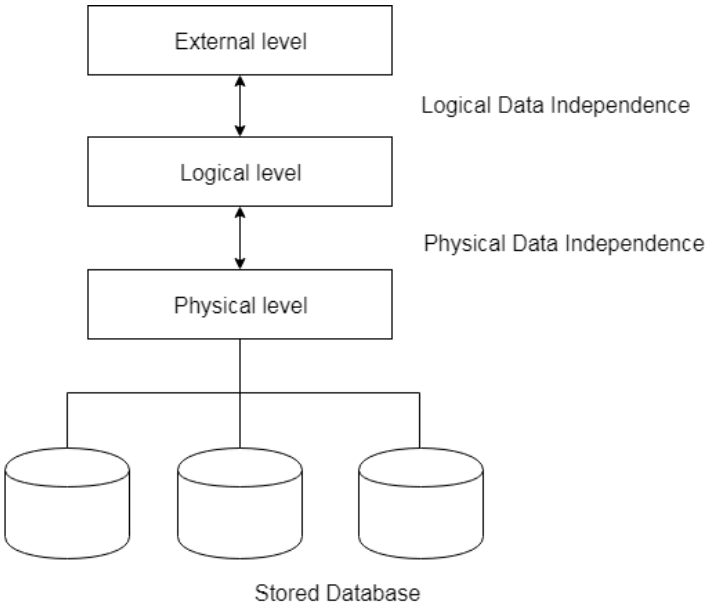


Fig: Data Independence

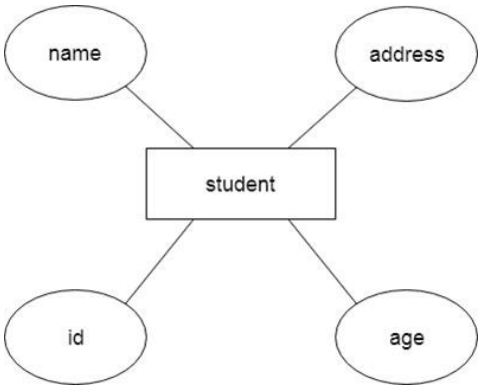
Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following four data models used for understanding the structure of the database:

Entity-Relationship Model(ER Model): It is a high-level data model which is used to define the data and the relationships between them. It is basically a conceptual design of any database which is easy to design the view of data.

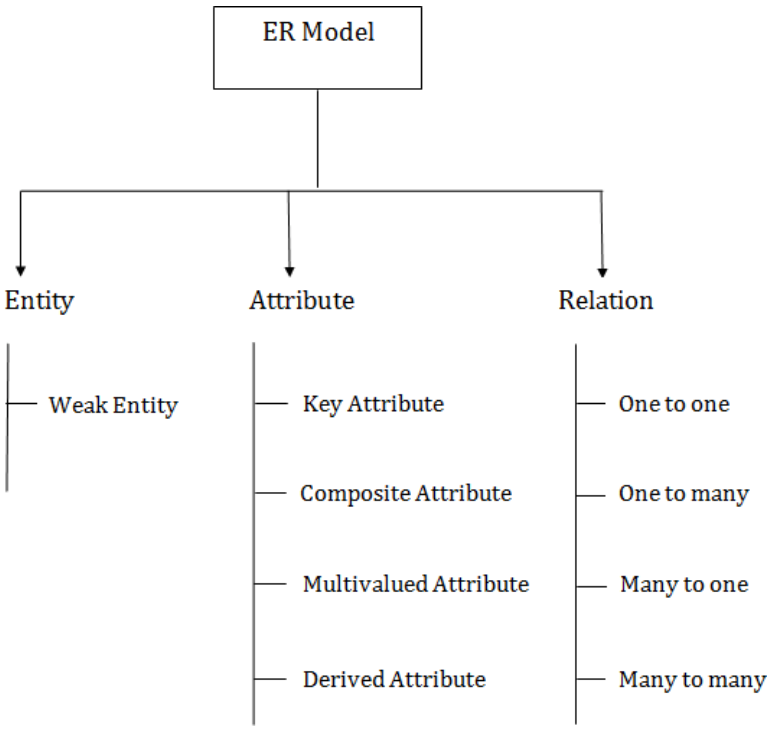
Why Use ER Diagrams In DBMS?

- ER diagrams represent the E-R model in a database, making them easy to convert into relations (tables).
- ER diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- ER diagrams require no technical knowledge and no hardware support.
- These diagrams are very easy to understand and easy to create even for a naive user.
- It gives a standard solution for visualizing the data logically.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



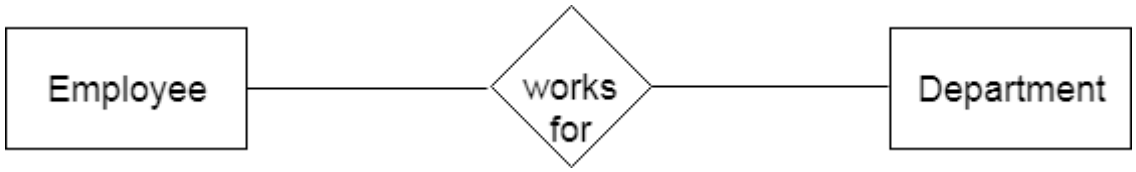
Component of ER Diagram



1. Entity:

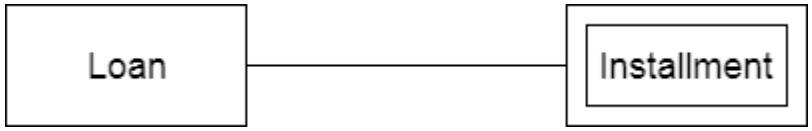
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

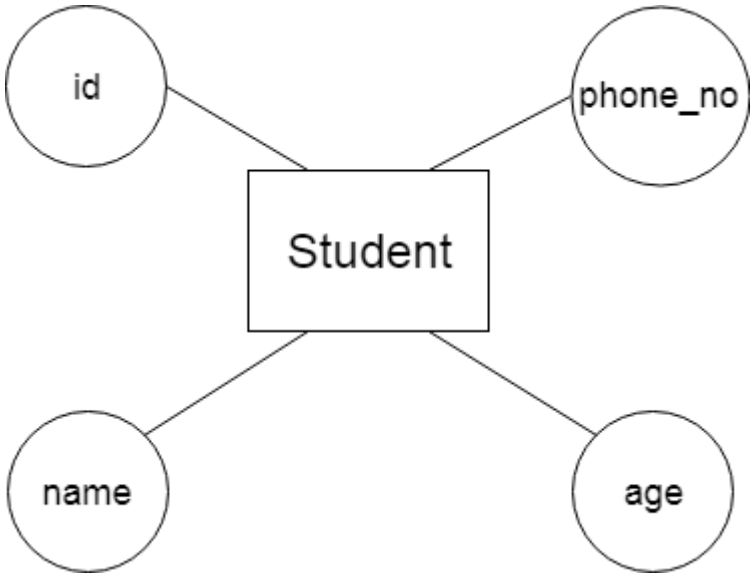
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

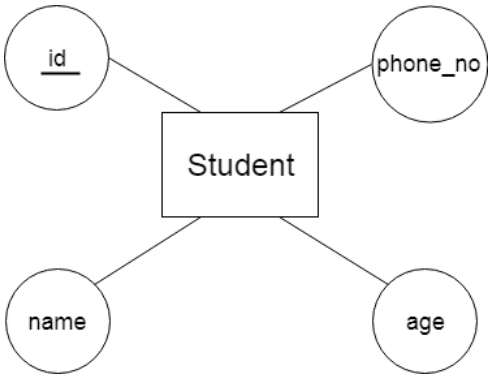
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



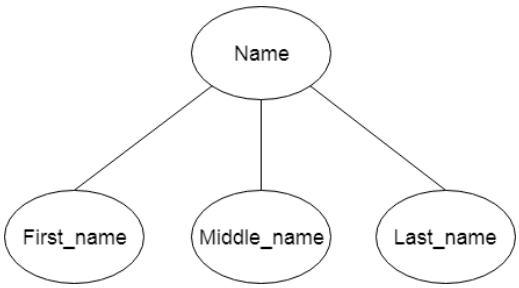
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

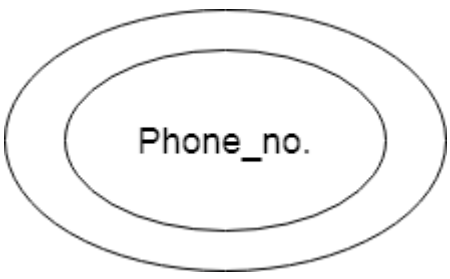
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

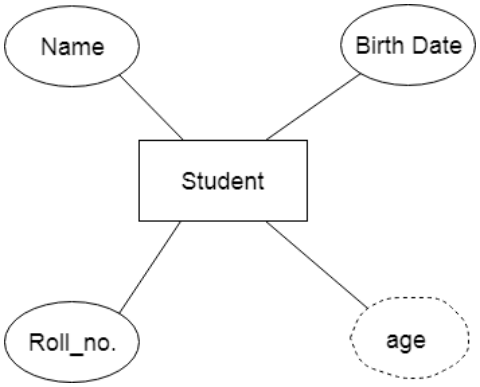
For example, a student can have more than one phone number.



d. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

Types of relationship are as follows:

a. One-to-One Relationship

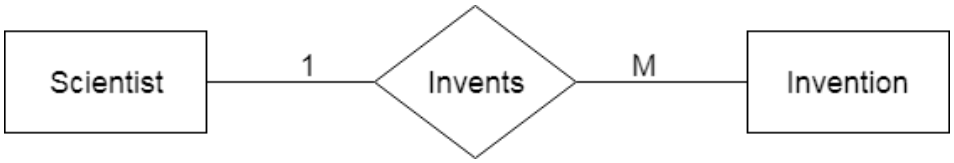
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

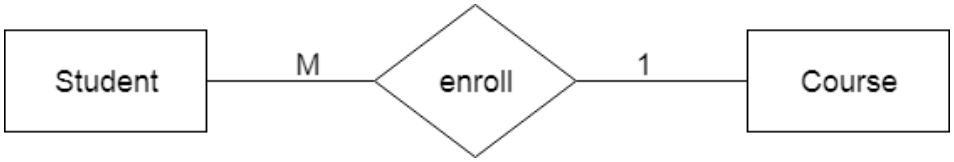
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



d. Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

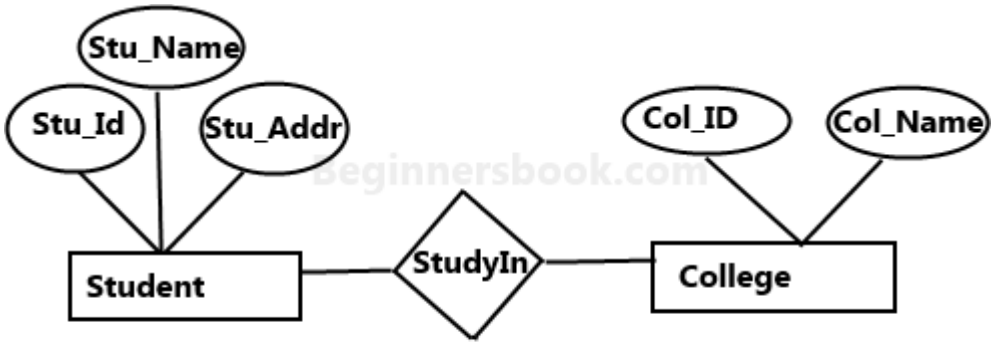
For example, Employee can assign by many projects and project can have many employees.

Entity Relationship Diagram (ER Diagram)

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute

of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Lets have a look at a simple ER diagram to understand this concept.

A simple ER Diagram:

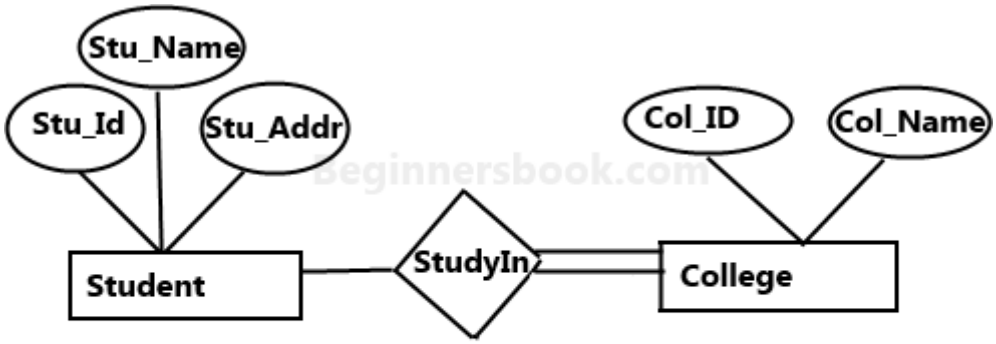


Sample E-R Diagram

In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.

Here are the geometric shapes and their meaning in an E-R Diagram. We will discuss these terms in detail in the next section(Components of a ER Diagram) of this guide so don't worry too much about these terms now, just go through them once.

Rectangle: Represents Entity sets.
Ellipses: Attributes
Diamonds: Relationship Set
Lines: They link attributes to Entity Sets and Entity sets to Relationship Set
Double Ellipses: Multivalued Attributes
Dashed Ellipses: Derived Attributes
Double Rectangles: Weak Entity Sets
Double Lines: Total participation of an entity in a relationship set



E-R Digram with total participation of College entity set in StudyIn relationship Set - This indicates that each college must have atleast one associated Student.

Relational Data Model: This type of model designs the data in the form of rows and columns within a table. Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations. This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

Example: STUDENT Relation

Stu_No	S_Name	PHONE_NO	ADDRESS	Gender
10112	Rama	9874567891	Islam ganj	F
12839	Shyam	9026288936	Delhi	M
33289	Laxman	8583287182	Gurugram	M
27857	Mahesh	7086819134	Ghaziabad	M
17282	Ganesh	9028939884	Delhi	M

Relation: A relation is usually represented as a table, organized into rows and columns. A relationship consists of multiple records. **For example:** student relation which contains tuples and attributes.

Tuple: The rows of a relation that contain the values corresponding to the attributes are called tuples. **For example:** in the Student relation there are 5 tuples.

The value of tuples contains (10112, Rama, 9874567891,islam ganj, F) etc.

Data Item: The smallest unit of data in the relation is the individual data item. It is stored at the intersection of rows and columns are also known as cells. **For Example:** 10112, "Rama" etc are data items in Student relation.

Domain: It contains a set of atomic values that an attribute can take. It could be accomplish explicitly by listing all possible values or specifying conditions that all values in that domain must be confirmed. **For example:** the domain of gender attributes is a set of data values "M" for male and "F" for female. No database software fully supports domains typically allowing the users to define very simple data types such as numbers, dates, characters etc.

Attribute: The smallest unit of data in relational model is an attribute. It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom(Ai). **For example:** Stu_No, S_Name, PHONE_NO, ADDRESS, Gender are the attributes of a student relation. In relational databases a column entry in any row is a single value that contains exactly one item only.

Cardinality: The total number of rows at a time in a relation is called the cardinality of that relation. For example: In a student relation, the total number of tuples in this relation is3 so the cardinality of a relation is 3. The cardinality of a relation changes with time as more and more tuples get added or deleted.

Degree: The degree of association is called the total number of attributes in a relationship. The relation with one attribute is called unary relation, with two attributes is known a binary relation and with three attributes is known as ternary relation. **For example:** in the Student relation, the total number of attributes is 5, so the degree of the relations is 5. The degree of a relation does not change with time as tuples get added or deleted.

Relational instance: In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples.

Relational schema: A relational schema contains the name of the relation and name of all columns or attributes.

Relational key: In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

Properties of Relations

- Each attribute in a relation has only one data value corresponding to it i.e. they do not contain two or more values.
- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name

- Attribute domain has no significance
- tuple has no duplicate value
- Order of tuple can have a different sequence
- It also provides information about metadata.

Object-based Data Model: An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types. Thus, in 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.

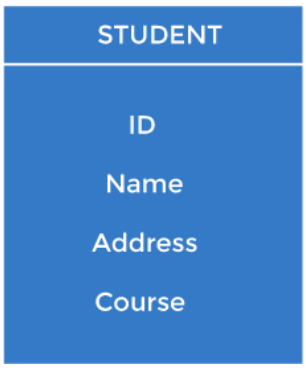
Hierarchical Model: The hierarchical Model is one of the oldest models in the data model which was developed by IBM, in the 1950s. In a hierarchical model, data are viewed as a collection of tables, or we can say segments that form a hierarchical relation. In this, the data is organized into a tree-like structure where each record consists of one parent record and many children. Even if the segments are connected as a chain-like structure by logical associations, then the instant structure can be a fan structure with multiple branches. We call the illogical associations as directional associations.

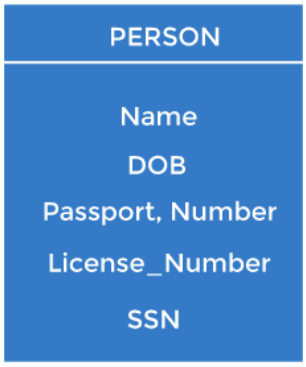
Network Model: The Network Model was formalized by the Database Task group in the 1960s. This model is the generalization of the hierarchical model. This model can consist of multiple parent segments and these segments are grouped as levels but there exists a logical association between the segments belonging to any level. Mostly, there exists a many-to-many logical association between any of the two segments.

Keys

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

For example, ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

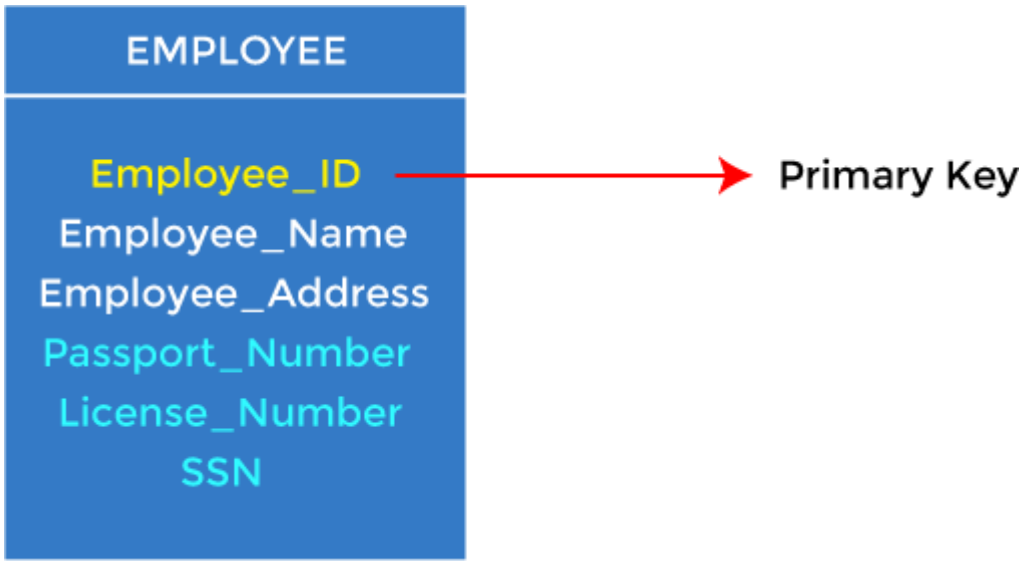




Types of keys:

1. Primary key

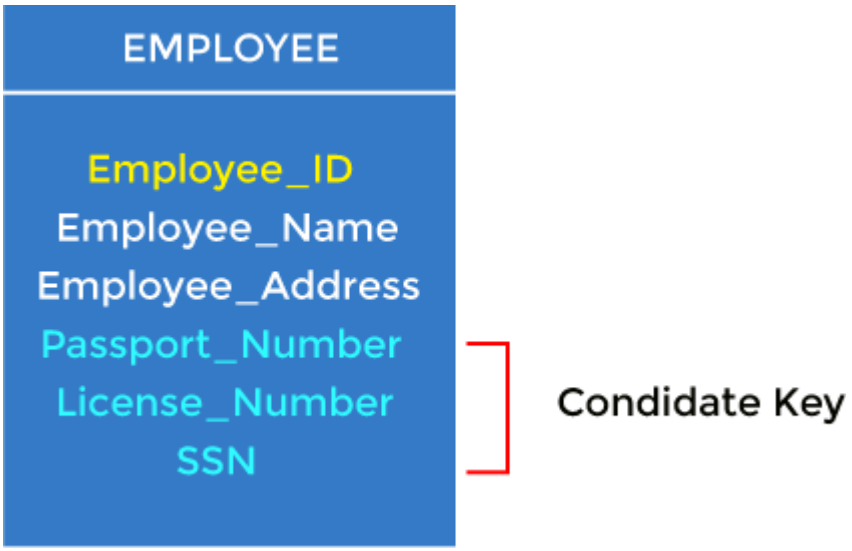
- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
- In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



2. Candidate key

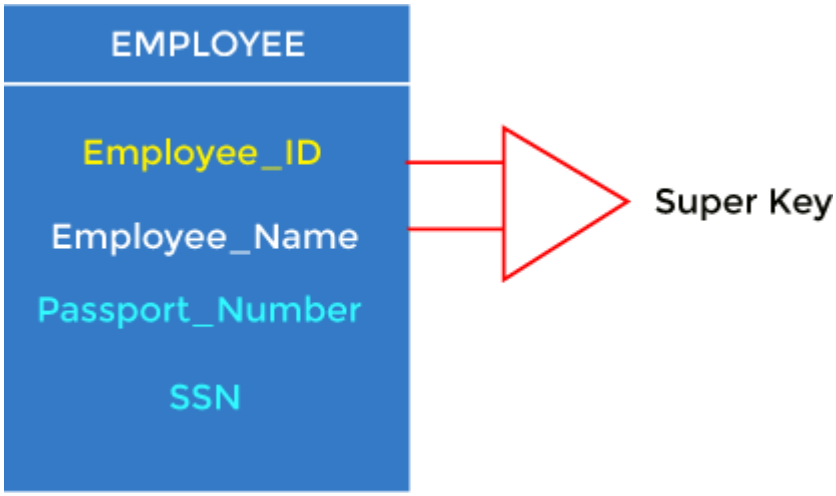
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Super Key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.

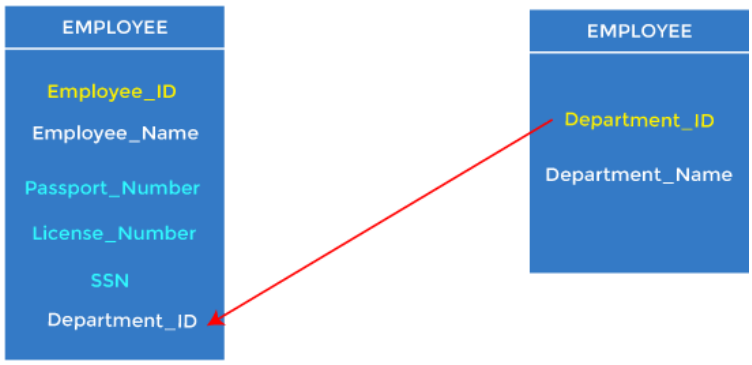


For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

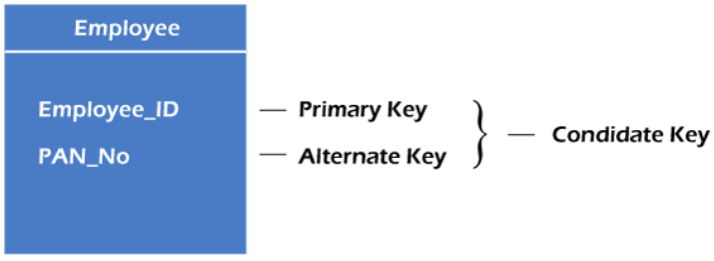
- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



5. Alternate key

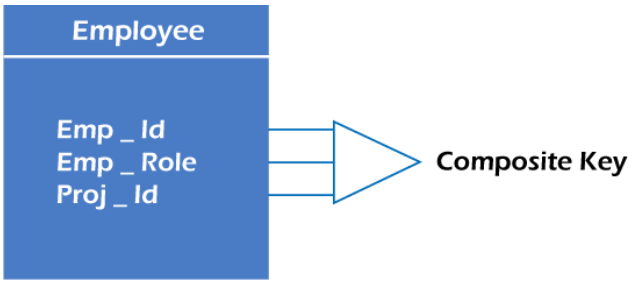
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. **In other words,** the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

For example, employee relation has two attributes, Employee_Id and PAN_No, that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.

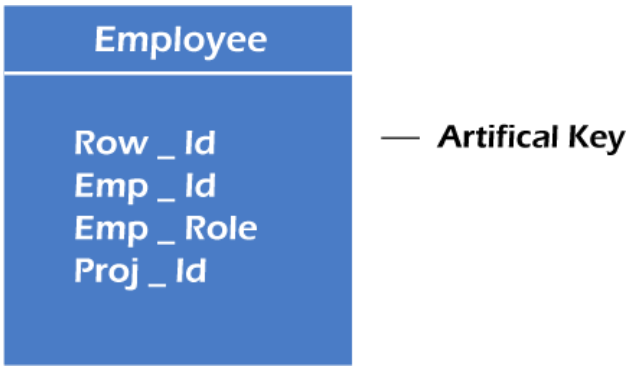


6. Composite key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



For example, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.



7. Artificial key

The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order.

For example, the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.

Constraints on Relational Database Model

In modeling the design of the relational database we can put some restrictions like what values are allowed to be inserted in the relation, and what kind of modifications and deletions are allowed in the relation. These are the restrictions we impose on the relational database. In models like Entity-Relationship models, we did not have such features. Database Constraints can be categorized into 3 main categories:

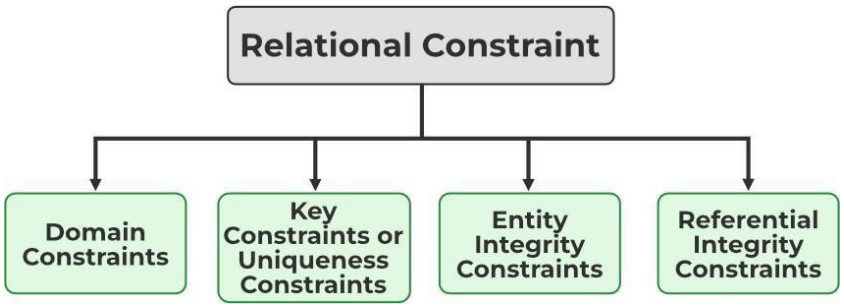
1. Constraints that are applied in the data model are called **Implicit Constraints**.
2. Constraints that are directly applied in the schemas of the data model, by specifying them in the DDL(Data Definition Language). These are called **Schema-Based Constraints or Explicit Constraints**.
3. Constraints that cannot be directly applied in the schemas of the data model. We call these Application-based or **Semantic Constraints**.

So here we are going to deal with **Implicit constraints**.

Relational Constraints

These are the restrictions or sets of rules imposed on the database contents. It validates the quality of the database. It validates the various operations like data insertion, updation, and other processes that have to be performed without affecting the integrity of the data. It protects us against threats/damages to the database. Mainly Constraints on the relational database are of 4 types

- Domain constraints
- Key constraints or Uniqueness Constraints
- Entity Integrity constraints
- Referential integrity constraints



Types of Relational Constraints

1. Domain Constraints

- Every domain must contain atomic values(smallest indivisible units) which means composite and multi-valued attributes are not allowed.
- We perform a datatype check here, which means when we assign a data type to a column we limit the values that it can contain. Eg. If we assign the datatype of attribute age as int, we can't give it values other than int datatype.

Example:

EID	**Name**	**Phone**
01	Bikash Dutta	123456789 234456678

Explanation: In the above relation, Name is a composite attribute and Phone is a multi-values attribute, so it is violating domain constraint.

2. Key Constraints or Uniqueness Constraints

- These are called uniqueness constraints since it ensures that every tuple in the relation should be unique.
- A relation can have multiple keys or candidate keys(minimal superkey), out of which we choose one of the keys as the primary key, we don't have any restriction on choosing the primary key out of candidate keys, but it is suggested to go with the candidate key with less number of attributes.
- Null values are not allowed in the primary key, hence Not Null constraint is also part of the key constraint.

Example:

EID	**Name**	**Phone**
01	Bikash	6000000009
02	Paul	9000090009
01	Tuhin	9234567892

Explanation: In the above table, EID is the primary key, and the first and the last tuple have the same value in EID ie 01, so it is violating the key constraint.

3. Entity Integrity Constraints

- Entity Integrity constraints say that no primary key can take a NULL value, since using the primary key we identify each tuple uniquely in a relation.

Example:

EID	**Name**	**Phone**
01	Bikash	9000900099
02	Paul	600000009
NULL	Sony	9234567892

Explanation: In the above relation, EID is made the primary key, and the primary key can't take NULL values but in the third tuple, the primary key is null, so it is violating Entity Integrity constraints.

4. Referential Integrity Constraints

- The Referential integrity constraint is specified between two relations or tables and used to maintain the consistency among the tuples in two relations.
- This constraint is enforced through a foreign key, when an attribute in the foreign key of relation R1 has the same domain(s) as the primary key of relation R2, then

the foreign key of R1 is said to reference or refer to the primary key of relation R2.

- The values of the foreign key in a tuple of relation R1 can either take the values of the primary key for some tuple in relation R2, or can take NULL values, but can't be empty.

Example:

EID	**Name**	**DNO**
01	Divine	12
02	Dino	22
04	Vivian	14

DNO	**Place**
12	Jaipur
13	Mumbai
14	Delhi

Explanation: In the above tables, the DNO of Table 1 is the foreign key, and DNO in Table 2 is the primary key. DNO = 22 in the foreign key of Table 1 is not allowed because DNO = 22 is not defined in the primary key of table 2. Therefore, Referential integrity constraints are violated here.