What is Data?

Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.

Word 'Data' is originated from the word 'datum' that means 'single piece of information.' It is plural of the word datum.

In computing, Data is information that can be translated into a form for efficient movement and processing. Data is interchangeable.

What is Database?

A **database** is an organized collection of data, so that it can be easily accessed and managed.

You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

Database handlers create a database in such a way that only one set of software program provides access of data to all the users.

The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many **dynamic websites** on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.

There are many **databases available** like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

A cylindrical structure is used to display the image of a database.

Evolution of Databases

The database has completed more than 50 years of journey of its evolution from flat-file system to relational and objects relational systems. It has gone through several generations.

The Evolution

File-Based

1968 was the year when File-Based database were introduced. In file-based databases, data was maintained in a flat file. Though files have many advantages, there are several limitations.

One of the major advantages is that the file system has various access methods, e.g., sequential, indexed, and random.

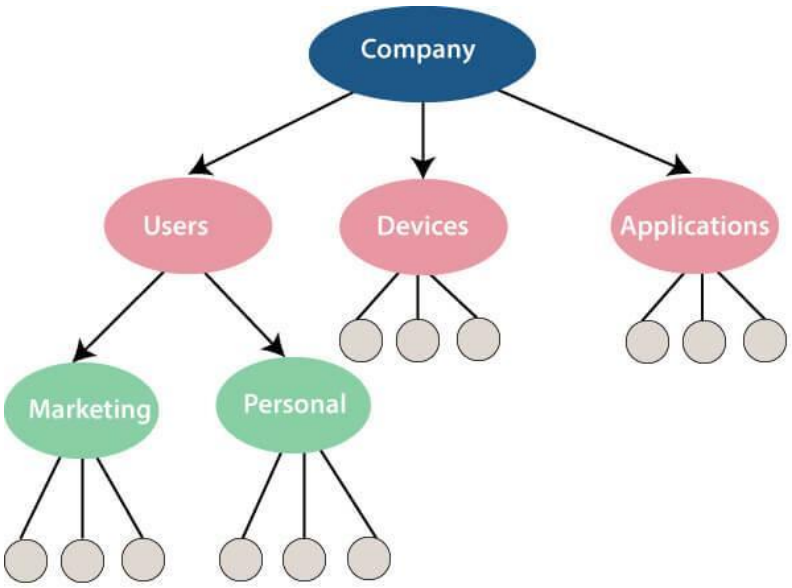
It requires extensive programming in a third-generation language such as COBOL, BASIC.

Hierarchical Data Model

1968-1980 was the era of the Hierarchical Database. Prominent hierarchical database model was IBM's first DBMS. It was called IMS (Information Management System).

In this model, files are related in a parent/child manner.

Below diagram represents Hierarchical Data Model. Small circle represents objects.



Like file system, this model also had some limitations like complex implementation, lack structural independence, can't easily handle a many-many relationship, etc.

Network data model

Charles Bachman developed the first DBMS at Honeywell called Integrated Data Store (IDS). It was developed in the early 1960s, but it was standardized in 1971 by the CODASYL group (Conference on Data Systems Languages).

In this model, files are related as owners and members, like to the common network model.

- Network schema (Database organization)
- Sub-schema (views of database per user)
- Data management language (procedural)

This model also had some limitations like system complexity and difficult to design and maintain.

Relational Database

1970 - Present: It is the era of Relational Database and Database Management. In 1970, the relational model was proposed by E.F. Codd.

Relational database model has two main terminologies called instance and schema.

The instance is a table with rows or columns

Schema specifies the structure like name of the relation, type of each column and name.

This model uses some mathematical concept like set theory and predicate logic.

The first internet database application had been created in 1995.

During the era of the relational database, many more models had introduced like object-oriented model, object-relational model, etc.

Cloud database

Cloud database facilitates you to store, manage, and retrieve their structured, unstructured data via a cloud platform. This data is accessible over the Internet. Cloud databases are also called a database as service (DBaaS) because they are offered as a managed service.

Some best cloud options are:

- AWS (Amazon Web Services)
- Snowflake Computing
- Oracle Database Cloud Services
- Microsoft SQL server
- Google cloud spanner

Advantages of cloud database

Lower costs

Generally, company provider does not have to invest in databases. It can maintain and support one or more data centers.

Automated

Cloud databases are enriched with a variety of automated processes such as recovery, failover, and auto-scaling.

Increased accessibility

You can access your cloud-based database from any location, anytime. All you need is just an internet connection.

NoSQL Database

A NoSQL database is an approach to design such databases that can accommodate a wide variety of data models. NoSQL stands for "not only SQL." It is an alternative to traditional relational

databases in which data is placed in tables, and data schema is perfectly designed before the database is built.

NoSQL databases are useful for a large set of distributed data.

Some examples of NoSQL database system with their category are:

- MongoDB, CouchDB, Cloudant **(Document-based)**
- Memcached, Redis, Coherence **(key-value store)**
- HBase, Big Table, Accumulo **(Tabular)**

Advantage of NoSQL

High Scalability

NoSQL can handle an extensive amount of data because of scalability. If the data grows, NoSQL database scale it to handle that data in an efficient manner.

High Availability

NoSQL supports auto replication. Auto replication makes it highly available because, in case of any failure, data replicates itself to the previous consistent state.

Disadvantage of NoSQL

Open source

NoSQL is an open-source database, so there is no reliable standard for NoSQL yet.

Management challenge

Data management in NoSQL is much more complicated than relational databases. It is very challenging to install and even more hectic to manage daily.

GUI is not available

GUI tools for NoSQL database are not easily available in the market.

Backup

Backup is a great weak point for NoSQL databases. Some databases, like MongoDB, have no powerful approaches for data backup.

DBMS (Data Base Management System)

Database management System is software which is used to store and retrieve the database. For example, Oracle, MySQL, etc.; these are some popular DBMS tools.

- DBMS provides the interface to perform the various operations like creation, deletion, modification, etc.
- DBMS allows the user to create their databases as per their requirement.
- DBMS accepts the request from the application and provides specific data through the operating system.
- DBMS contains the group of programs which acts according to the user instruction.
- It provides security to the database.

The advantages of the DBMS are explained below –

Redundancy problem can be solved.

In the File System, duplicate data is created in many places because all the programs have their own files which create data redundancy resulting in wastage of memory. In DBMS, all the files are integrated in a single database. So there is no chance of duplicate data.

For example: A student record in a library or examination can contain duplicate values, but when they are converted into a single database, all the duplicate values are removed.

Has a very high security level.

Data security level is high by protecting your precious data from unauthorized access. Only authorized users should have the grant to access the database with the help of credentials.

Presence of Data integrity.

Data integrity makes unification of so many files into a single file. DBMS allows data integrity which makes it easy to decrease data duplicity Data integration and reduces redundancy as well as data inconsistency.

Support multiple users.

DBMS allows multiple users to access the same database at a time without any conflicts.

Avoidance of inconsistency.

DBMS controls data redundancy and also controls data consistency. Data consistency is nothing but if you want to update data in any files then all the files should not be updated again.

In DBMS, data is stored in a single database so data becomes more consistent in comparison to file processing systems.

Shared data

Data can be shared between authorized users of the database in DBMS. All the users have their own right to access the database. Admin has complete access to the database. He has a right to assign users to access the database.

Enforcement of standards

As DBMS have central control of the database. So, a DBA can ensure that all the applications follow some standards such as format of data, document standards etc. These standards help in data migrations or in interchanging the data.

Any unauthorized access is restricted

Unauthorized persons are not allowed to access the database because of security credentials.

Provide backup of data

Data loss is a big problem for all the organizations. In the file system users have to back up the files in regular intervals which lead to waste of time and resources.

DBMS solves this problem of taking backup automatically and recovery of the database.

Tunability

Tuning means adjusting something to get a better performance. Same in the case of DBMS, as it provides tunability to improve performance. DBA adjusts databases to get effective results.

Disadvantages of DBMS

The disadvantages of DBMS are as follows:

Complexity

The provision of the functionality that is expected of a good DBMS makes the DBMS an extremely complex piece of software. Database designers, developers, database administrators and end-users must understand this functionality to take full advantage of it.

Failure to understand the system can lead to bad design decisions, which leads to a serious consequence for an organization.

Size

The functionality of DBMS makes use of a large piece of software which occupies megabytes of disk space.

Performance

Performance may not run as fast as desired.

Higher impact of a failure

The centralization of resources increases the vulnerability of the system because all users and applications rely on the availability of DBMS, the failure of any component can bring operation to halt.

Cost of DBMS

The cost of DBMS varies significantly depending on the environment and functionality provided. There is also the recurrent annual maintenance cost.

RDBMS (Relational Database Management System)

The word RDBMS is termed as 'Relational Database Management System.' It is represented as a table that contains rows and column.

RDBMS is based on the Relational model; it was introduced by E. F. Codd.

A relational database contains the following components:

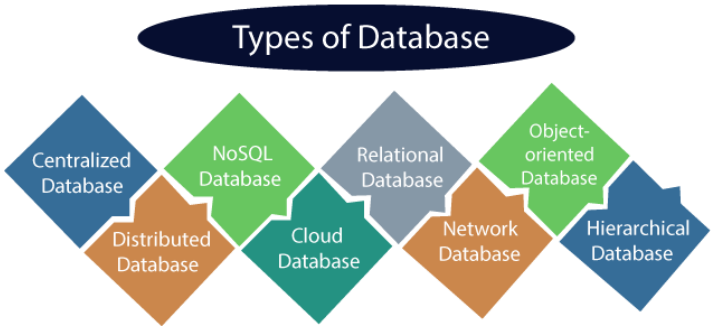
- Table
- Record/ Tuple
- Field/Column name /Attribute
- Instance

- Schema
- Keys

An RDBMS is a tabular DBMS that maintains the security, integrity, accuracy, and consistency of the data.

Types of Databases

There are various types of databases used for storing different varieties of data:



1) Centralized Database

It is the type of database that stores data at a centralized database system. It comforts the users to access the stored data from different locations through several applications. These applications contain the authentication process to let users access data securely. An example of a Centralized database can be Central Library that carries a central database of each library in a college/university.

Advantages of Centralized Database

- It has decreased the risk of data management, i.e., manipulation of data will not affect the core data.
- Data consistency is maintained as it manages data in a central repository.
- It provides better data quality, which enables organizations to establish data standards.
- It is less costly because fewer vendors are required to handle the data sets.

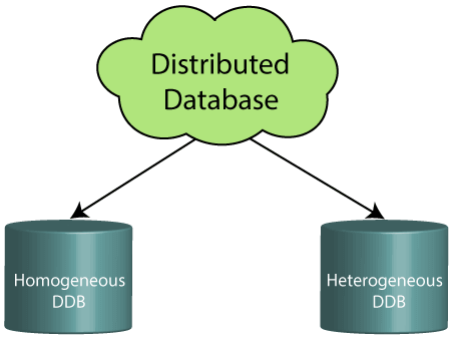
Disadvantages of Centralized Database

- The size of the centralized database is large, which increases the response time for fetching the data.
- It is not easy to update such an extensive database system.
- If any server failure occurs, entire data will be lost, which could be a huge loss.

2) Distributed Database

Unlike a centralized database system, in distributed systems, data is distributed among different database systems of an organization. These database systems are connected via communication links. Such links help the end-users to access the data easily. **Examples** of the Distributed database are Apache Cassandra, HBase, Ignite, etc.

We can further divide a distributed database system into:



- **Homogeneous DDB:** Those database systems which execute on the same operating system and use the same application process and carry the same hardware devices.
- **Heterogeneous DDB:** Those database systems which execute on different operating systems under different application procedures, and carries different hardware devices.

Advantages of Distributed Database

- Modular development is possible in a distributed database, i.e., the system can be expanded by including new computers and connecting them to the distributed system.
- One server failure will not affect the entire data set.

3) Relational Database

This database is based on the relational data model, which stores data in the form of rows(tuple) and columns(attributes), and together forms a table(relation). A relational database uses SQL for storing, manipulating, as well as maintaining the data. E.F. Codd invented the database in 1970. Each table in the database carries a key that makes the data unique from others. **Examples** of Relational databases are MySQL, Microsoft SQL Server, Oracle, etc.

Properties of Relational Database

There are following four commonly known properties of a relational model known as ACID properties, where:

A means Atomicity: This ensures the data operation will complete either with success or with failure. It follows the 'all or nothing' strategy. For example, a transaction will either be committed or will abort.

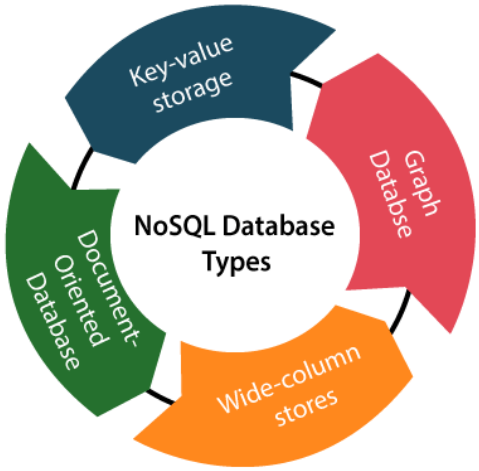
C means Consistency: If we perform any operation over the data, its value before and after the operation should be preserved. For example, the account balance before and after the transaction should be correct, i.e., it should remain conserved.

I means Isolation: There can be concurrent users for accessing data at the same time from the database. Thus, isolation between the data should remain isolated. For example, when multiple transactions occur at the same time, one transaction effects should not be visible to the other transactions in the database.

D means Durability: It ensures that once it completes the operation and commits the data, data changes should remain permanent.

NoSQL Database

Non-SQL/Not Only SQL is a type of database that is used for storing a wide range of data sets. It is not a relational database as it stores data not only in tabular form but in several different ways. It came into existence when the demand for building modern applications increased. Thus, NoSQL presented a wide variety of database technologies in response to the demands. We can further divide a NoSQL database into the following four types:



. **Key-value storage:** It is the simplest type of database storage where it stores every single item as a key (or attribute name) holding its value, together.

. **Document-oriented Database:** A type of database used to store data as JSON-like document. It helps developers in storing data by using the same document-model format as used in the application code.

. **Graph Databases:** It is used for storing vast amounts of data in a graph-like structure. Most commonly, social networking websites use the graph database.

. **Wide-column stores:** It is similar to the data represented in relational databases. Here, data is stored in large columns together, instead of storing in rows.

Advantages of NoSQL Database

- It enables good productivity in the application development as it is not required to store data in a structured format.
- It is a better option for managing and handling large data sets.
- It provides high scalability.
- Users can quickly access data from the database through key-value.

5) Cloud Database

A type of database where data is stored in a virtual environment and executes over the cloud computing platform. It provides users with various cloud computing services (SaaS, PaaS, IaaS, etc.) for accessing the database. There are numerous cloud platforms, but the best options are:

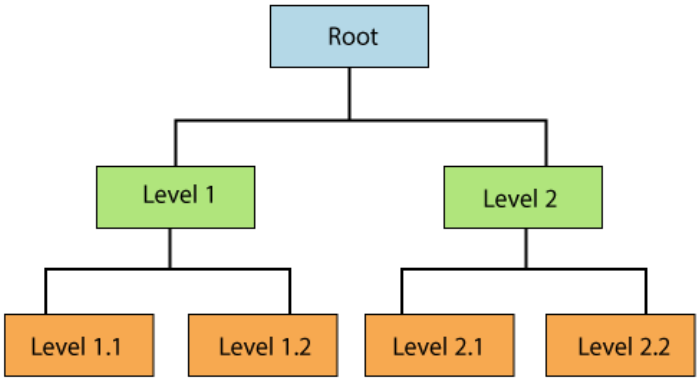
- Amazon Web Services(AWS)
- Microsoft Azure
- Kamatera
- PhonixNAP
- ScienceSoft
- Google Cloud SQL, etc.

6) Object-oriented Databases

The type of database that uses the object-based data model approach for storing data in the database system. The data is represented and stored as objects which are similar to the objects used in the object-oriented programming language.

7) Hierarchical Databases

It is the type of database that stores data in the form of parent-children relationship nodes. Here, it organizes data in a tree-like structure.



Hierarchical Database

Data get stored in the form of records that are connected via links. Each child record in the tree will contain only one parent. On the other hand, each parent record can have multiple child records.

8) Network Databases

It is the database that typically follows the network data model. Here, the representation of data is in the form of nodes connected via links between them. Unlike the hierarchical database, it allows each record to have multiple children and parent nodes to form a generalized graph structure.

9) Personal Database

Collecting and storing data on the user's system defines a Personal Database. This database is basically designed for a single user.

Advantage of Personal Database

- It is simple and easy to handle.
- It occupies less storage space as it is small in size.

10) Operational Database

The type of database which creates and updates the database in real-time. It is basically designed for executing and handling the daily data operations in several businesses. For example, An organization uses operational databases for managing per day transactions.

11) Enterprise Database

Large organizations or enterprises use this database for managing a massive amount of data. It helps organizations to increase and improve their efficiency. Such a database allows simultaneous access to users.

Advantages of Enterprise Database:

- Multi processes are supportable over the Enterprise database.
- It allows executing parallel queries on the system.

DBMS Applications

There are different fields where a database management system is utilized. Following are a few applications which utilize the information base administration framework –

1. **Railway Reservation System –**
   In the rail route reservation framework, the information base is needed to store the record or information of ticket appointments, status about train's appearance, and flight. Additionally, if trains get late, individuals become acquainted with it through the information base update.

2. **Library Management System –**
   There are lots of books in the library so; it is difficult to store the record of the relative multitude of books in a register or duplicate. Along these lines, the data set administration framework (DBMS) is utilized to keep up all the data identified with the name of the book, issue date, accessibility of the book, and its writer.

3. **Banking –**
   Database the executive's framework is utilized to store the exchange data of the client in the information base.

4. **Education Sector –**
   Presently, assessments are led online by numerous schools and colleges. They deal with all assessment information through the data set administration framework (DBMS). In spite of that understudy's enlistments subtleties, grades, courses, expense, participation, results, and so forth all the data is put away in the information base.

5. **Credit card exchanges –**
   The database Management framework is utilized for buying on charge cards and age of month to month proclamations.

6. **Social Media Sites –**
   We all utilization of online media sites to associate with companions and to impart our perspectives to the world. Every day, many people group pursue these online media accounts like Pinterest, Facebook, Twitter, and Google in addition to. By the utilization of the data set administration framework, all the data of clients are put away in the information base and, we become ready to

interface with others.

7. **Broadcast communications –**
   Without DBMS any media transmission organization can't think. The Database the executive's framework is fundamental for these organizations to store the call subtleties and month to month postpaid bills in the information base.

8. **Account –**
   The information base administration framework is utilized for putting away data about deals, holding and acquisition of monetary instruments, for example, stocks and bonds in a data set.

9. **Online Shopping –**
   These days, web-based shopping has become a major pattern. Nobody needs to visit the shop and burn through their time. Everybody needs to shop through web based shopping sites, (for example, Amazon, Flipkart, Snapdeal) from home. So all the items are sold and added uniquely with the assistance of the information base administration framework (DBMS). Receipt charges, installments, buy data these are finished with the assistance of DBMS.

10. **Human Resource Management –**
    Big firms or organizations have numerous specialists or representatives working under them. They store data about worker's compensation, assessment, and work with the assistance of an information base administration framework (DBMS).

11. **Manufacturing –**
    Manufacturing organizations make various kinds of items and deal them consistently. To keep the data about their items like bills, acquisition of the item, amount, inventory network the executives, information base administration framework (DBMS) is utilized.

12. **Airline Reservation System –**
    This framework is equivalent to the railroad reservation framework. This framework additionally utilizes an information base administration framework to store the records of flight takeoff, appearance, and defer status.
13. **Healthcare:** DBMS is used in healthcare to manage patient data, medical records, and billing information.

Characteristics of DBMS:

1. **Data retrieval:** DBMS provides a way to retrieve data quickly and easily using search queries.

2. **Data manipulation:** DBMS provides tools to manipulate data, such as sorting, filtering, and aggregating data.
3. **Security:** DBMS provides security features to ensure that only authorized users have access to the data.
4. **Data backup and recovery:** DBMS provides tools to back up data and recover it in case of system failures or data loss.
5. **Multi-user access:** DBMS allows multiple users to access and modify data simultaneously.
6. **Reporting and analysis:** DBMS provides tools to generate reports and analyze data to gain insights and make informed decisions.

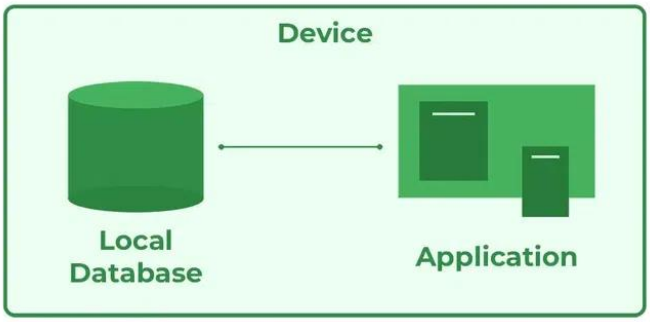
DBMS Architecture

- 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
There are several types of DBMS Architecture that we use according to the usage requirements. Types of DBMS Architecture are discussed here.

- 1-Tier Architecture
- 2-Tier Architecture
- 3-Tier Architecture

1-Tier Architecture
In 1-Tier Architecture the database is directly available to the user, the user can directly sit on the DBMS and use it that is, the client, server, and Database are all present on the same machine. For Example: to learn SQL we set up an SQL server and the database on the local system. This enables us to directly interact with the relational database and execute operations. The industry won't use this architecture they logically go for 2-tier and 3-tier Architecture.



DBMS 1-Tier Architecture

Advantages of 1-Tier Architecture

Below mentioned are the advantages of 1-Tier Architecture.

- **Simple Architecture:** 1-Tier Architecture is the most simple architecture to set up, as only a single machine is required to maintain it.
- **Cost-Effective:** No additional hardware is required for implementing 1-Tier Architecture, which makes it cost-effective.
- **Easy to Implement:** 1-Tier Architecture can be easily deployed, and hence it is mostly used in small projects.

2-Tier Architecture

The 2-tier architecture is similar to a basic client-server model. The application at the client end directly communicates with the database on the server side. APIs like ODBC and JDBC are used for this interaction. The server side is responsible for providing query processing and transaction management functionalities. On the client side, the user interfaces and application programs are run. The application on the client side establishes a connection with the server side to communicate with the DBMS.

An advantage of this type is that maintenance and understanding are easier, and compatible with existing systems. However, this model gives poor performance when there are a large number of users.

DBMS 2-Tier Architecture

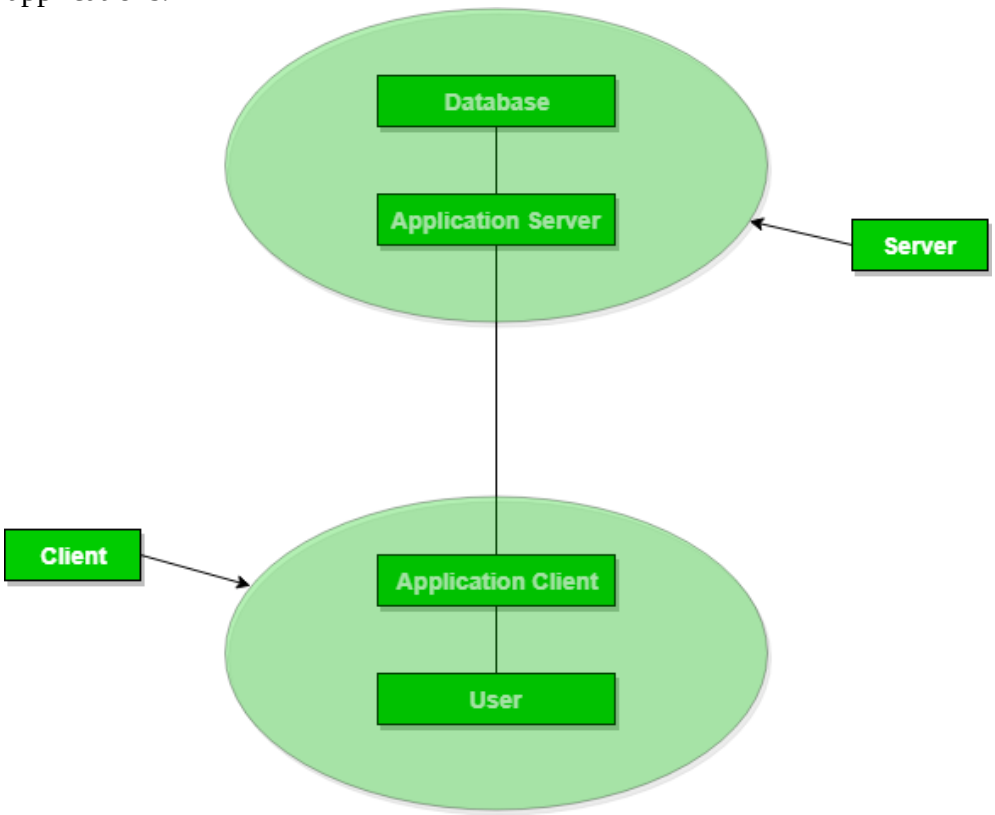
Advantages of 2-Tier Architecture

- **Easy to Access:** 2-Tier Architecture makes easy access to the database, which makes fast retrieval.

- **Scalable:** We can scale the database easily, by adding clients or upgrading hardware.
- **Low Cost:** 2-Tier Architecture is cheaper than 3-Tier Architecture and Multi-Tier Architecture.
- **Easy Deployment:** 2-Tier Architecture is easier to deploy than 3-Tier Architecture.
- **Simple:** 2-Tier Architecture is easily understandable as well as simple because of only two components.

3-Tier Architecture

In 3-Tier Architecture, there is another layer between the client and the server. The client does not directly communicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place. This intermediate layer acts as a medium for the exchange of partially processed data between the server and the client. This type of architecture is used in the case of large web applications.



DBMS 3-Tier Architecture

Advantages of 3-Tier Architecture

- **Enhanced scalability:** Scalability is enhanced due to the distributed deployment of application servers. Now, individual connections need not be made between the client and server.
- **Data Integrity:** 3-Tier Architecture maintains Data Integrity. Since there is a middle layer between the client and the server, data corruption can be avoided/removed.
- **Security:** 3-Tier Architecture Improves Security. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.

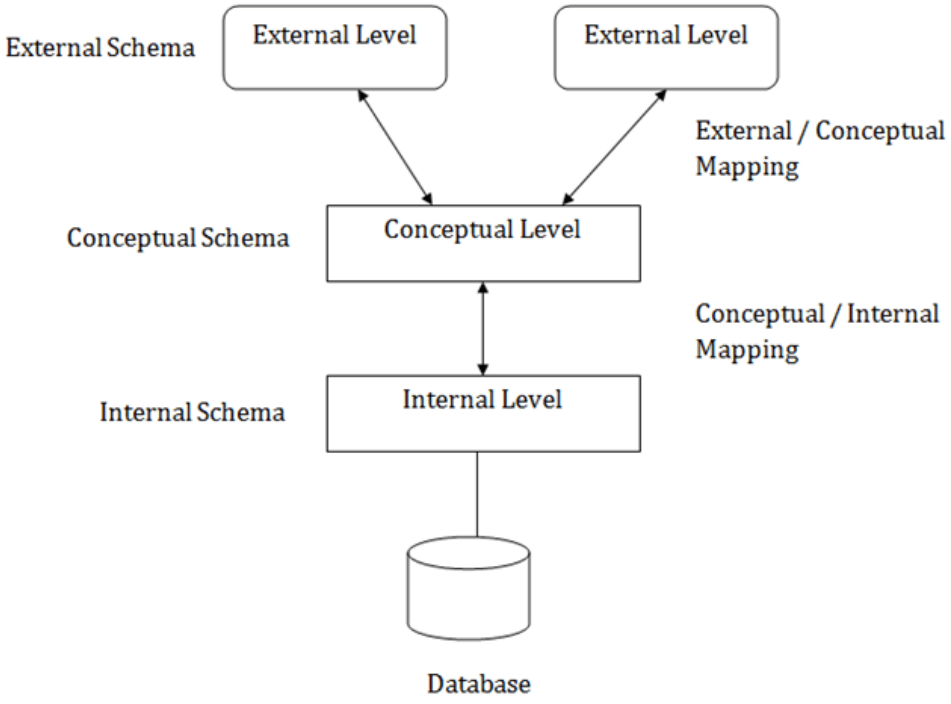
Disadvantages of 3-Tier Architecture

- **More Complex:** 3-Tier Architecture is more complex in comparison to 2-Tier Architecture. Communication Points are also doubled in 3-Tier Architecture.
- **Difficult to Interact:** It becomes difficult for this sort of interaction to take place due to the presence of middle layers.

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:

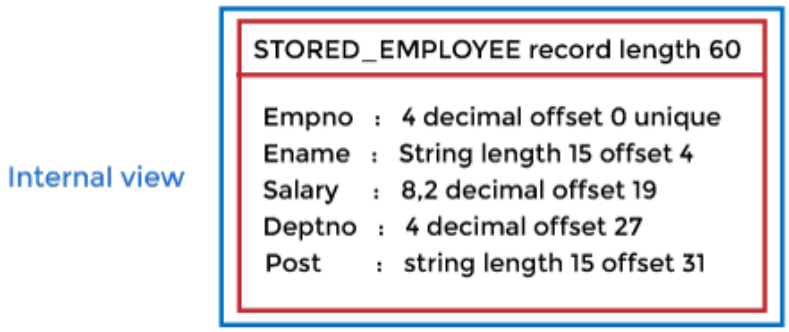
- It shows the DBMS architecture.
- Mapping is used to transform the request and response between various database levels of architecture.
- Mapping is not good for small DBMS because it takes more time.
- In External / Conceptual mapping, it is necessary to transform the request from external level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

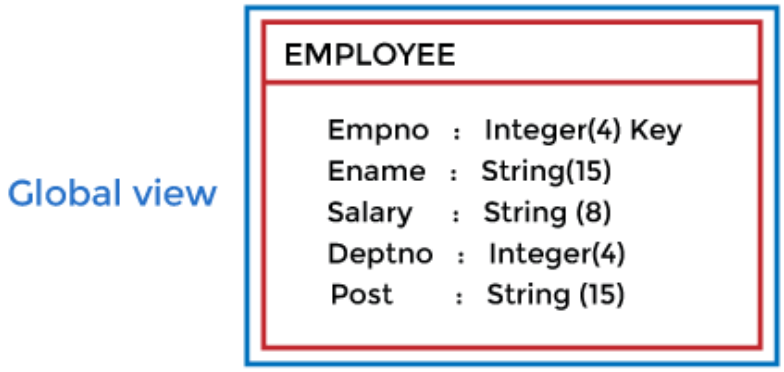


- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.
- It uses the physical data model. It is used to define that how the data will be stored in a block.
- The physical level is used to describe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

Mapping between Views

The three levels of DBMS architecture don't exist independently of each other. There must be correspondence between the three levels i.e. how they actually correspond with each other. DBMS is responsible for correspondence between the three types of schema. This correspondence is called Mapping.

There are basically two types of mapping in the database architecture:

- Conceptual/ Internal Mapping
- External / Conceptual Mapping

Conceptual/ Internal Mapping

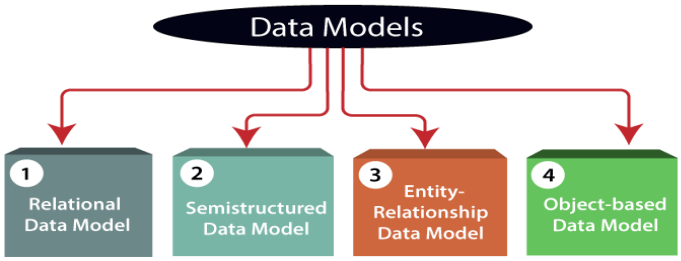
The Conceptual/ Internal Mapping lies between the conceptual level and the internal level. Its role is to define the correspondence between the records and fields of the conceptual level and files and data structures of the internal level.

External/ Conceptual Mapping

The external/Conceptual Mapping lies between the external level and the Conceptual level. Its role is to define the correspondence between a particular external and the conceptual view.

Data Models

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:



1) 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.

2) Entity-Relationship Data Model: An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and relationship is an association among these entities. This model was designed by Peter Chen and published in 1976 papers. It was widely used in database designing. A set of attributes describe the entities. For example, student_name, student_id describes the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

3) 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.

4) Semistructured Data Model: This type of data model is different from the other three data models (explained above). The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets. The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data.

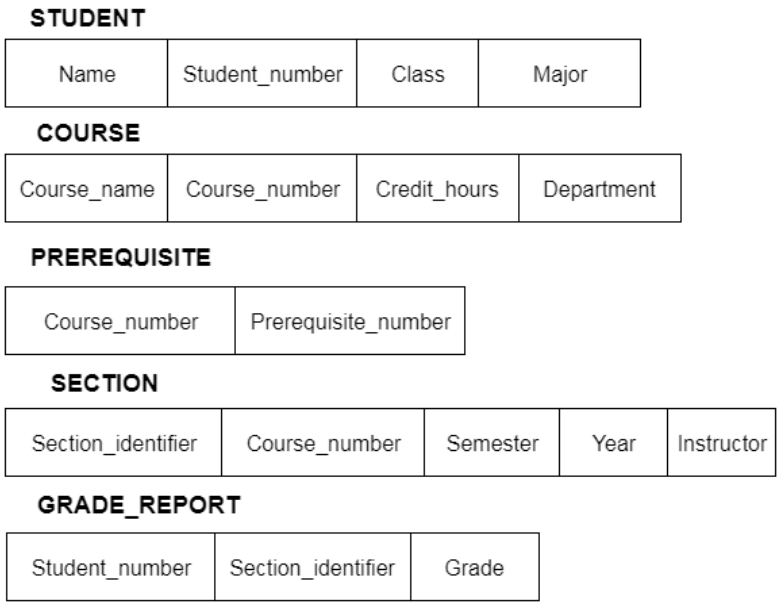
Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

Data model Schema and Instance

- The data which is stored in the database at a particular moment of time is called an instance of the database.
- The overall design of a database is called schema.
- A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- A database schema can be represented by using the visual diagram. That diagram shows the database objects and relationship with each other.
- A database schema is designed by the database designers to help programmers whose software will interact with the database. The process of database creation is called data modeling.

A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema diagram. For example, the given figure neither show the data type of each data item nor the relationship among various files.

In the database, actual data changes quite frequently. For example, in the given figure, the database changes whenever we add a new grade or add a student. The data at a particular moment of time is called the instance of the database.



Data Independence

- Data independence can be explained using the three-schema architecture.
- Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

There are two types of data independence:

1. Logical Data Independence

- Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
- Logical data independence is used to separate the external level from the conceptual view.
- If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.
- Logical data independence occurs at the user interface level.

2. **Physical Data Independence**

- Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.

- If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
- Physical data independence is used to separate conceptual levels from the internal levels.
- Physical data independence occurs at the logical interface level.

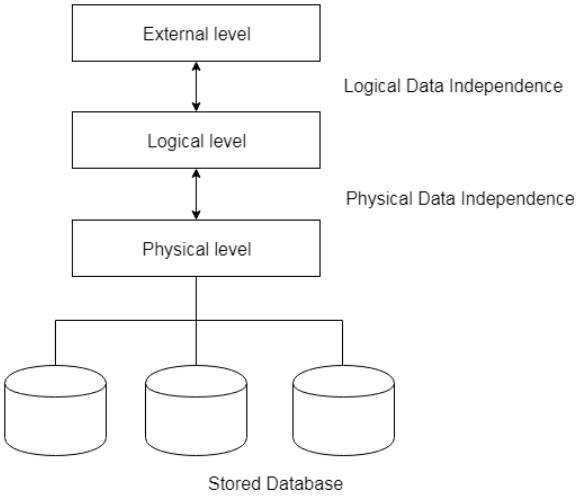
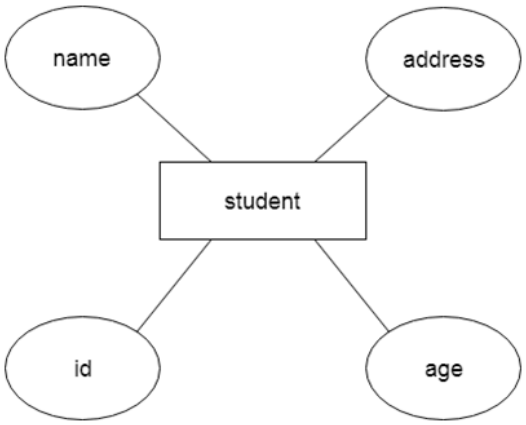


Fig: Data Independence

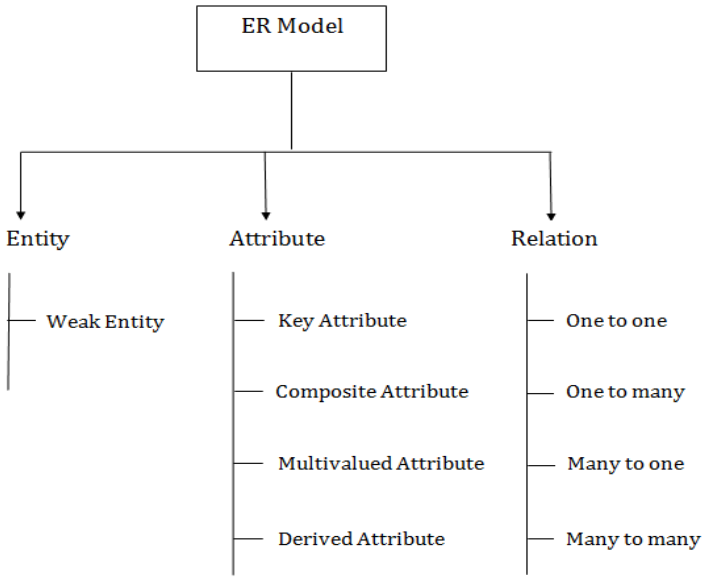
ER (Entity Relationship) Diagram in DBMS

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

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



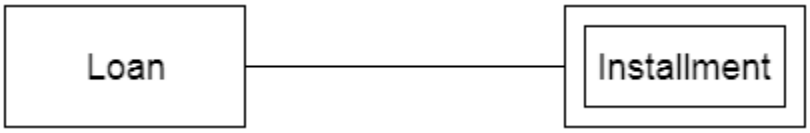
1. Entity:

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

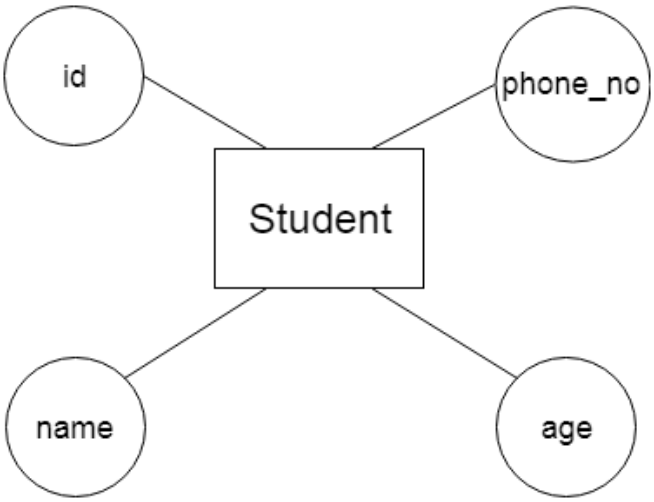
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

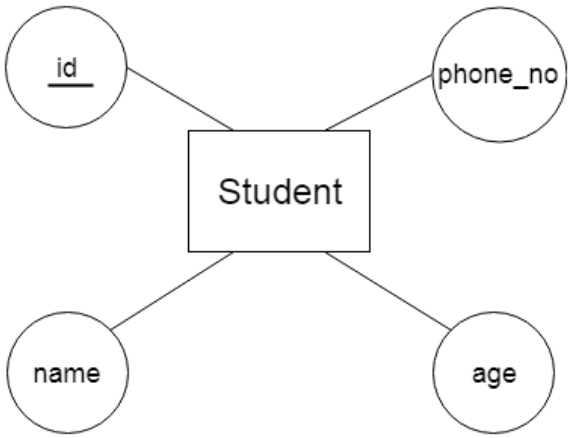
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

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



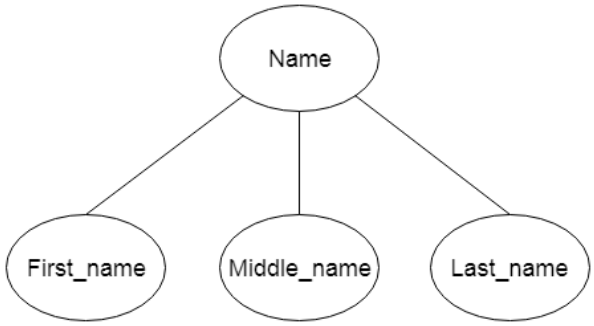
a. Key Attribute

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

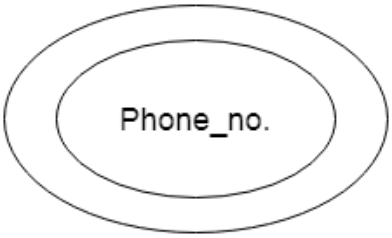
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



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.

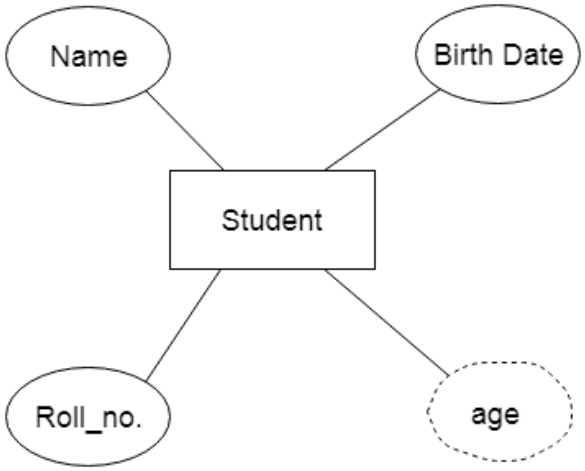
For example, a student can have more than one phone number.



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

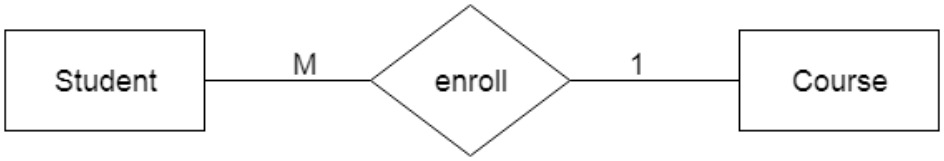
When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

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.

Notation of ER diagram

Database can be represented using the notations. In ER diagram, many notations are used to express the cardinality. These notations are as follows:

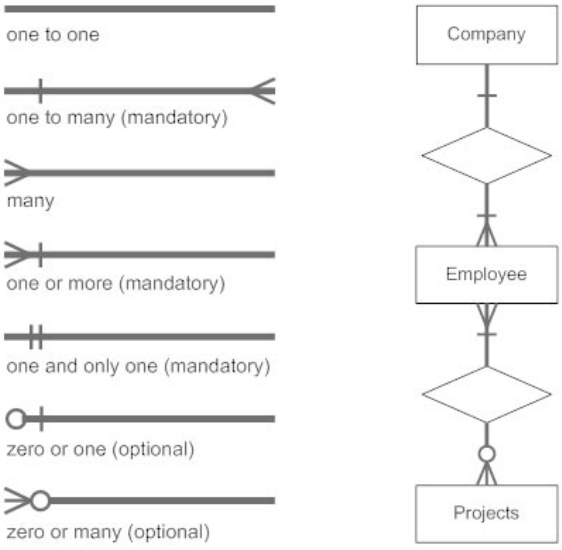


Fig: Notations of ER diagram

Cardinality in DBMS (Mapping Constraints)

DBMS

DBMS stands for Database Management System, which is a tool, or a software used to do various operations on a Database like the Creation of the Database, Deletion of the Database, or Updating the current Database. To simplify processing and data querying, the most popular types of Databases currently in use typically model their data as rows and columns in a set of tables. The data may then be handled, updated, regulated, and structured with ease. For writing and querying data, most Databases employ Structured Query Language (SQL).

Cardinality

Cardinality means how the entities are arranged to each other or what is the relationship structure between entities in a relationship set. In a Database Management System, Cardinality represents a number that denotes how many times an entity is participating with another entity in a relationship set. The Cardinality of DBMS is a very important attribute in representing the structure of a Database. In a table, the number of rows or tuples represents the Cardinality.

Cardinality Ratio

Cardinality ratio is also called **Cardinality Mapping**, which represents the mapping of one entity set to another entity set in a relationship set. We generally take the example of a binary relationship set where two entities are mapped to each other.

Cardinality is very important in the Database of various businesses. For example, if we want to track the purchase history of each customer then we can use the one-to-many cardinality to find the data of a specific customer. The Cardinality model can be used in Databases by Database Managers for a variety of purposes, but corporations often use it to evaluate customer or inventory data.

here are four types of Cardinality Mapping in Database Management Systems:

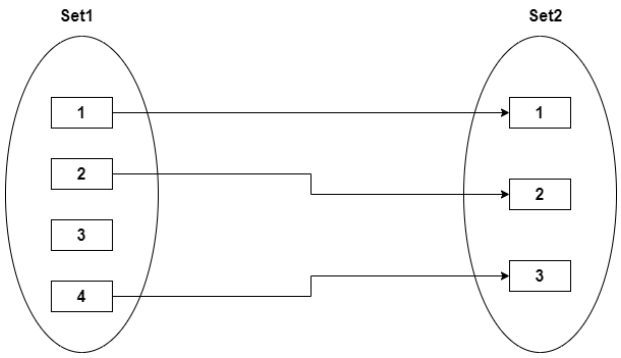
1. One to one
2. Many to one
3. One to many
4. Many to many

One to One

One to one cardinality is represented by a **1:1** symbol. In this, there is at most one relationship from one entity to another entity. There are a lot of examples of one-to-one cardinality in real life databases.

For example, one student can have only one student id, and one student id can belong to only one student. So, the relationship mapping between student and student id will be one to one cardinality mapping.

Another example is the relationship between the director of the school and the school because one school can have a maximum of one director, and one director can belong to only one school.

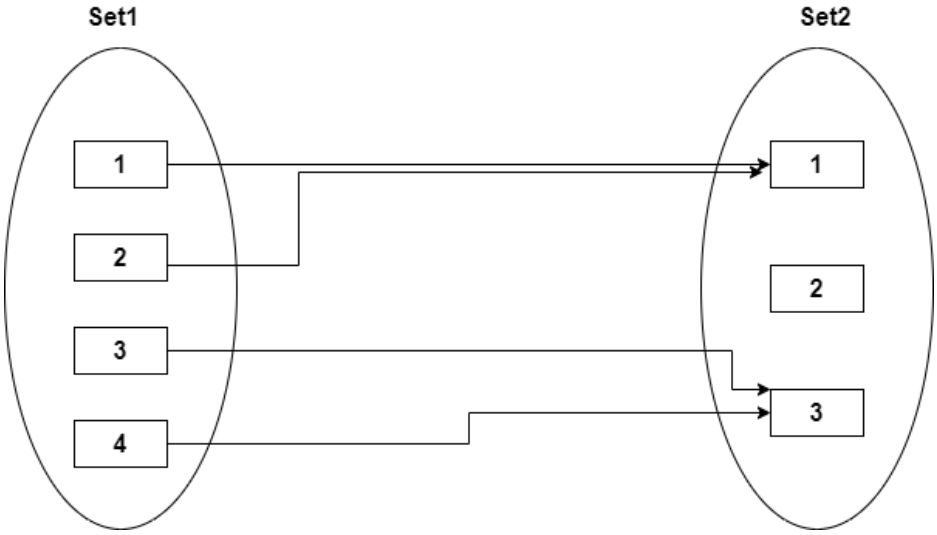


Many to One Cardinality:

In many to one cardinality mapping, from set 1, there can be multiple sets that can make relationships with a single entity of set 2. Or we can also describe it as from set 2, and one entity can make a relationship with more than one entity of set 1.

One to one Cardinality is the subset of Many to one Cardinality. It can be represented by **M:1**.

For example, there are multiple patients in a hospital who are served by a single doctor, so the relationship between patients and doctors can be represented by Many to one Cardinality.

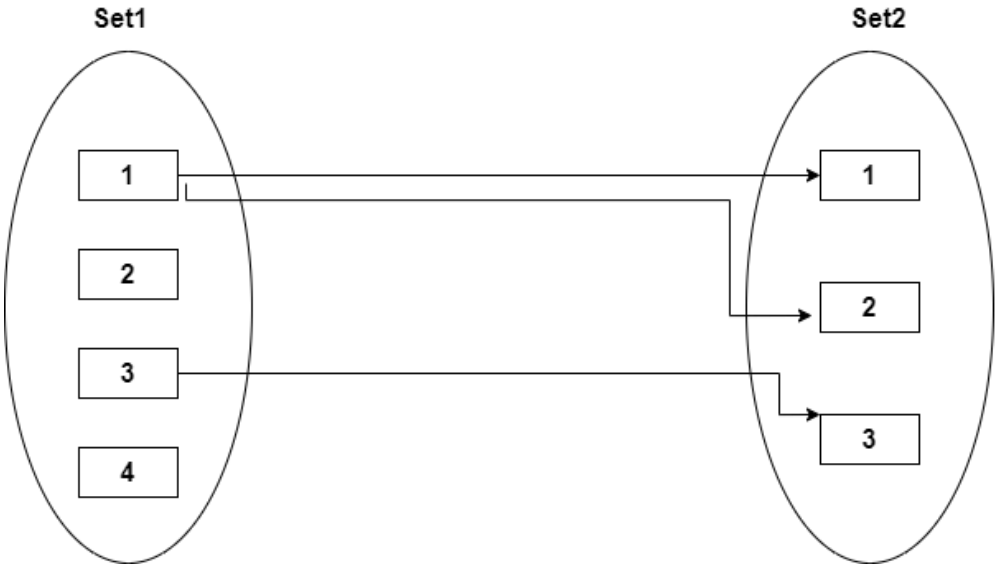


One to Many Cardinalities:

In One-to-many cardinality mapping, from set 1, there can be a maximum single set that can make relationships with a single or more than one entity of set 2. Or we can also describe it as from set 2, more than one entity can make a relationship with only one entity of set 1.

One to one cardinality is the subset of One-to-many Cardinality. It can be represented by **1: M**.

For Example, in a hospital, there can be various compounders, so the relationship between the hospital and compounders can be mapped through One-to-many Cardinality.



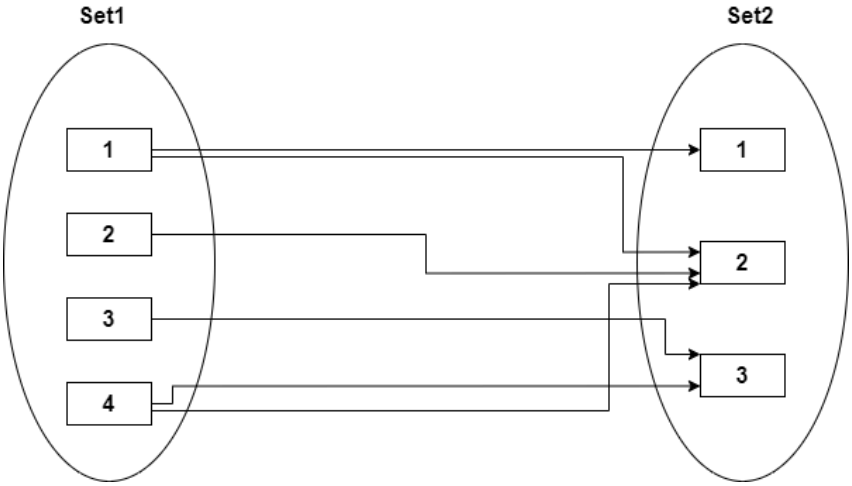
Many to Many Cardinalities:

In many, many cardinalities mapping, there can be one or more than one entity that can associate with one or more than one entity of set 2. In the same way from the end of set 2, one or more than one entity can make a relation with one or more than one entity of set 1.

It is represented by **M: N** or **N: M**.

One to one cardinality, One to many cardinalities, and Many to one cardinality is the subset of the many to many cardinalities.

For Example, in a college, multiple students can work on a single project, and a single student can also work on multiple projects. So, the relationship between the project and the student can be represented by many to many cardinalities.



Appropriate Mapping Cardinality

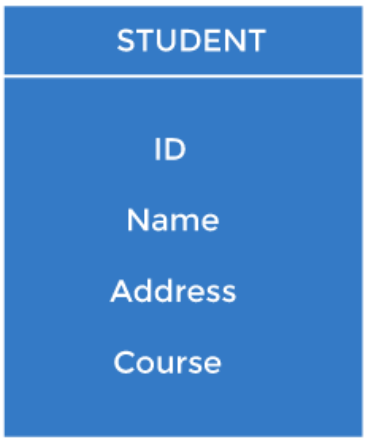
Evidently, the real-world context in which the relation set is modeled determines the Appropriate Mapping Cardinality for a specific relation set.

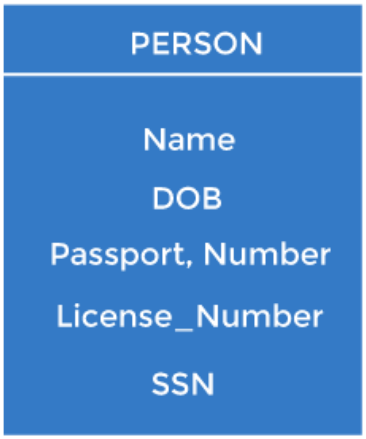
- We can combine relational tables with many involved tables if the Cardinality is one-to-many or many-to-one.
- One entity can be combined with a relation table if it has a one-to-one relationship and total participation, and two entities can be combined with their relation to form a single table if both of them have total participation.
- We cannot mix any two tables if the Cardinality is many-to-many.

Keys

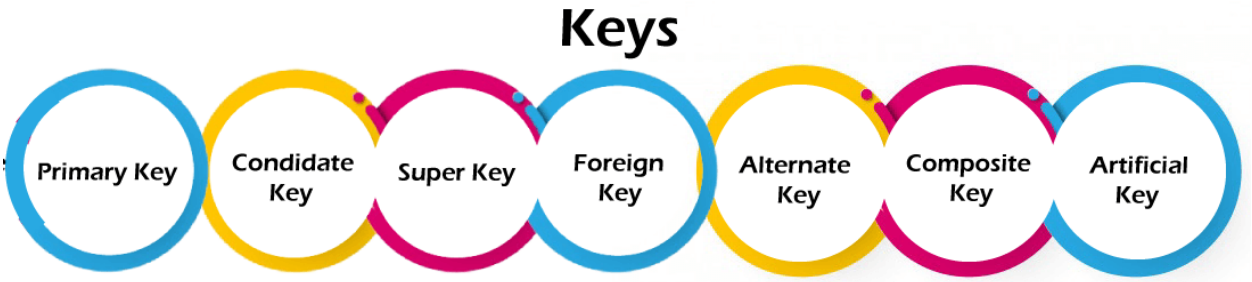
- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

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.



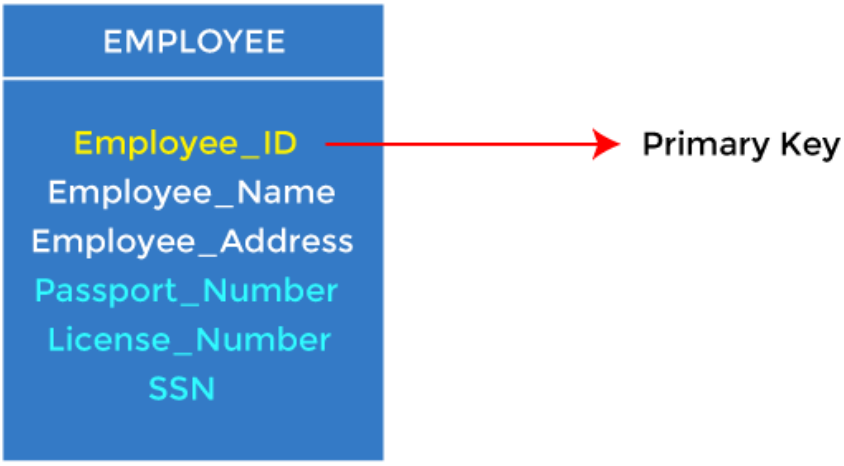


Types of keys:



1. Primary key

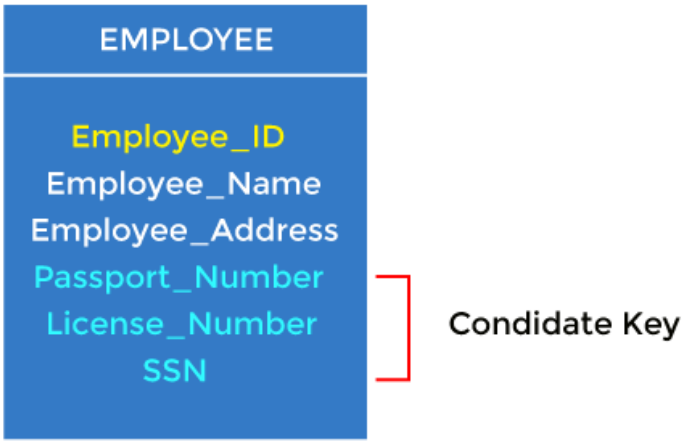
- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
- In the EMPLOYEE table, ID can be the primary key since it is unique for 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

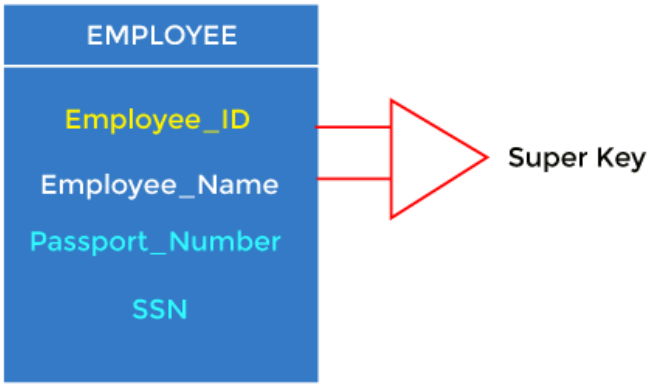
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

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.

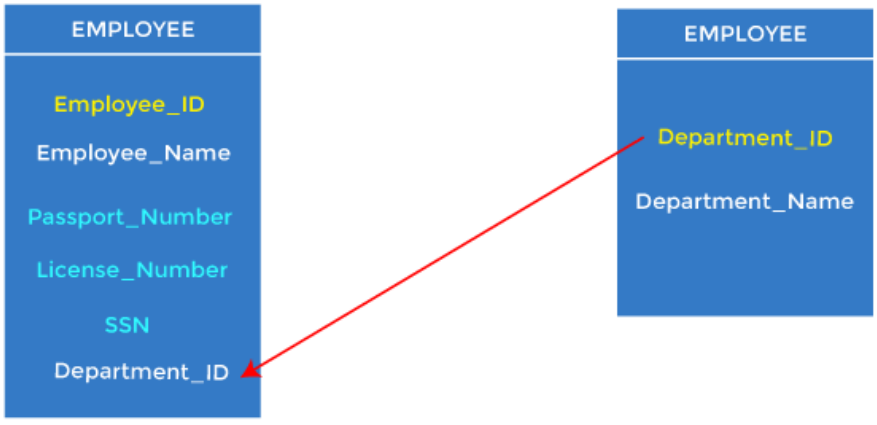


For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

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.

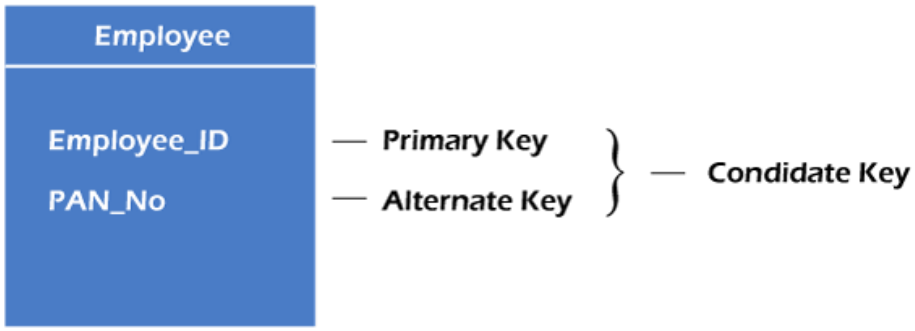


5. Alternate key

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

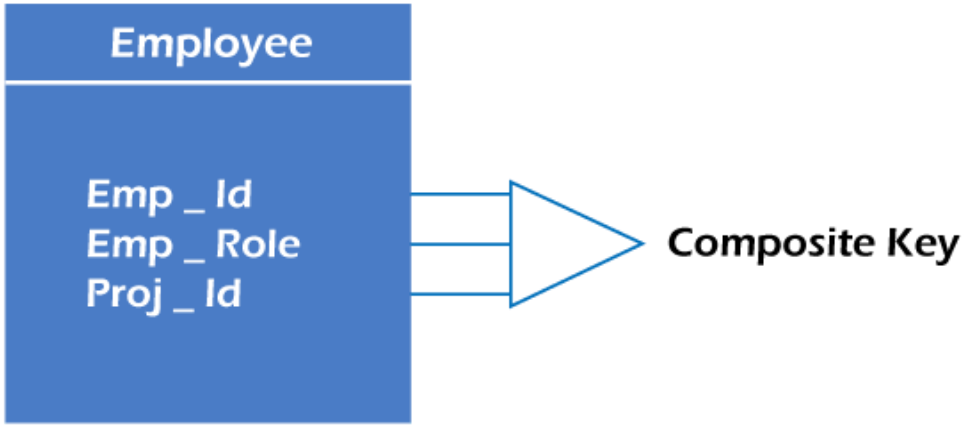
the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

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.

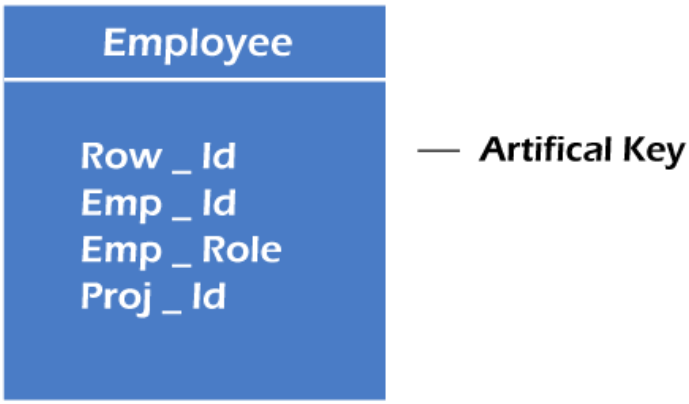


6. Composite key

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

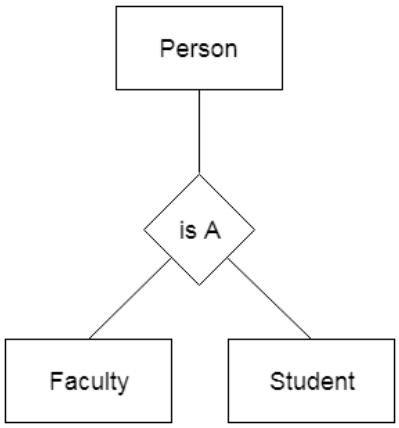
The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order.

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.

Generalization

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

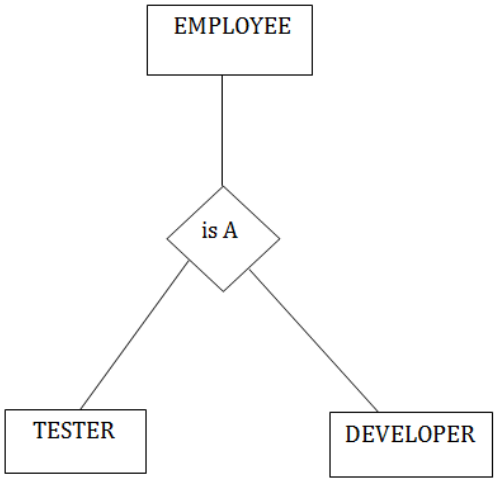
For example, Faculty and Student entities can be generalized and create a higher level entity Person.



Specialization

- Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
- Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

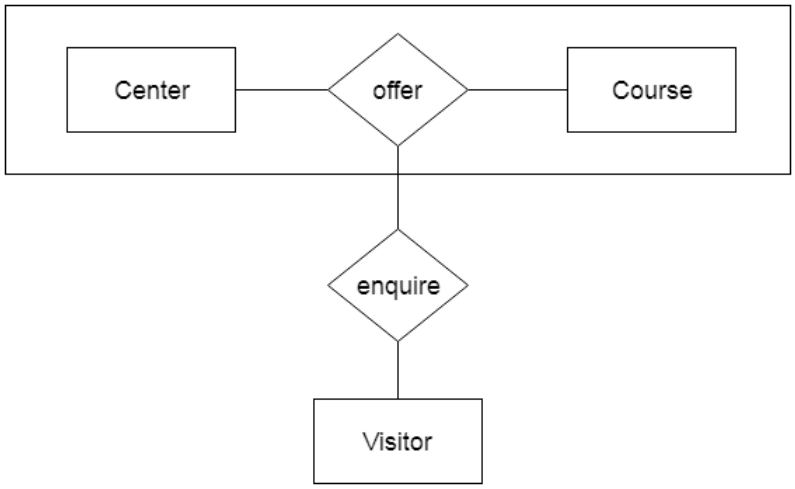
For example: In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



Aggregation

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

For example: Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.

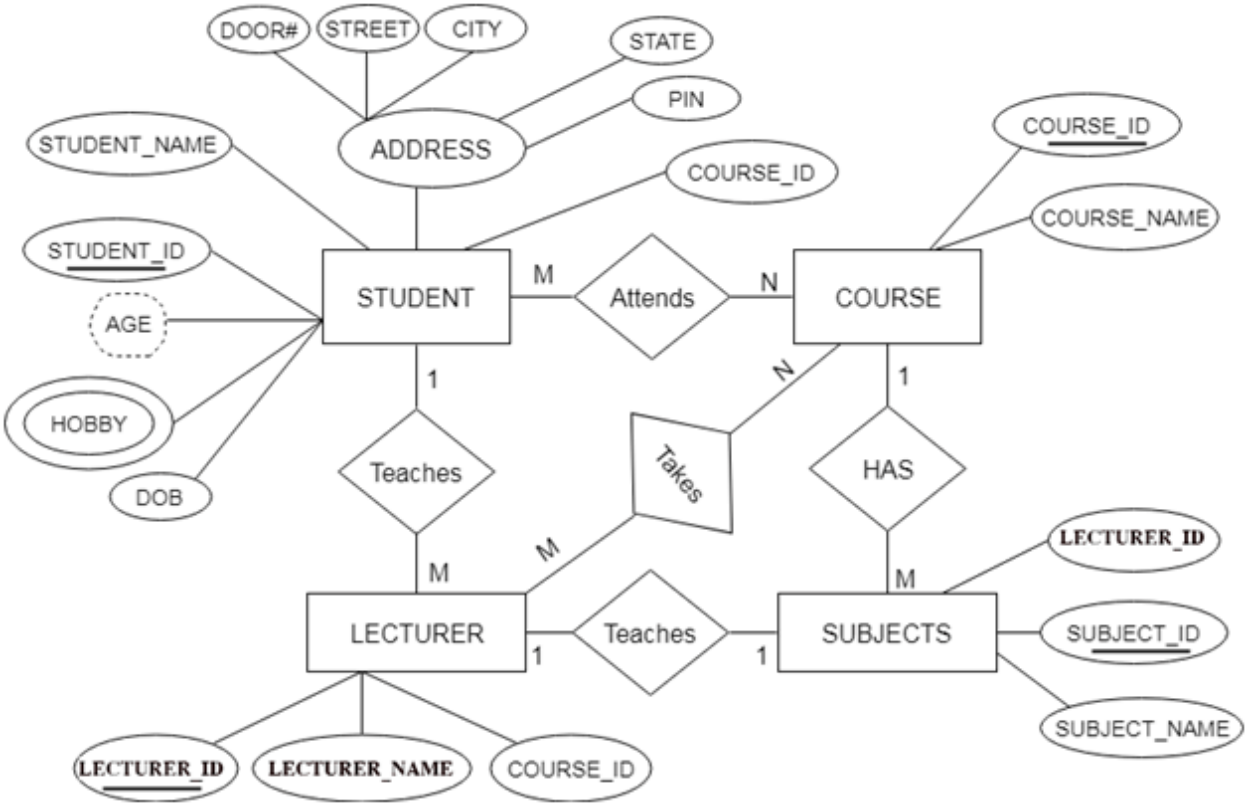


Reduction of ER diagram to Table

The database can be represented using the notations, and these notations can be reduced to a collection of tables.

In the database, every entity set or relationship set can be represented in tabular form.

The ER diagram is given below:



There are some points for converting the ER diagram to the table:

- **Entity type becomes a table.**

In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.

- **All single-valued attribute becomes a column for the table.**

In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table. Similarly, COURSE_NAME and COURSE_ID form the column of COURSE table and so on.

- **A key attribute of the entity type represented by the primary key.**

In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID, and LECTURE_ID are the key attribute of the entity.

- **The multivalued attribute is represented by a separate table.**

In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both the column, we create a composite key.

- **Composite attribute represented by components.**

In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

- **Derived attributes are not considered in the table.**

In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

Using these rules, you can convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:

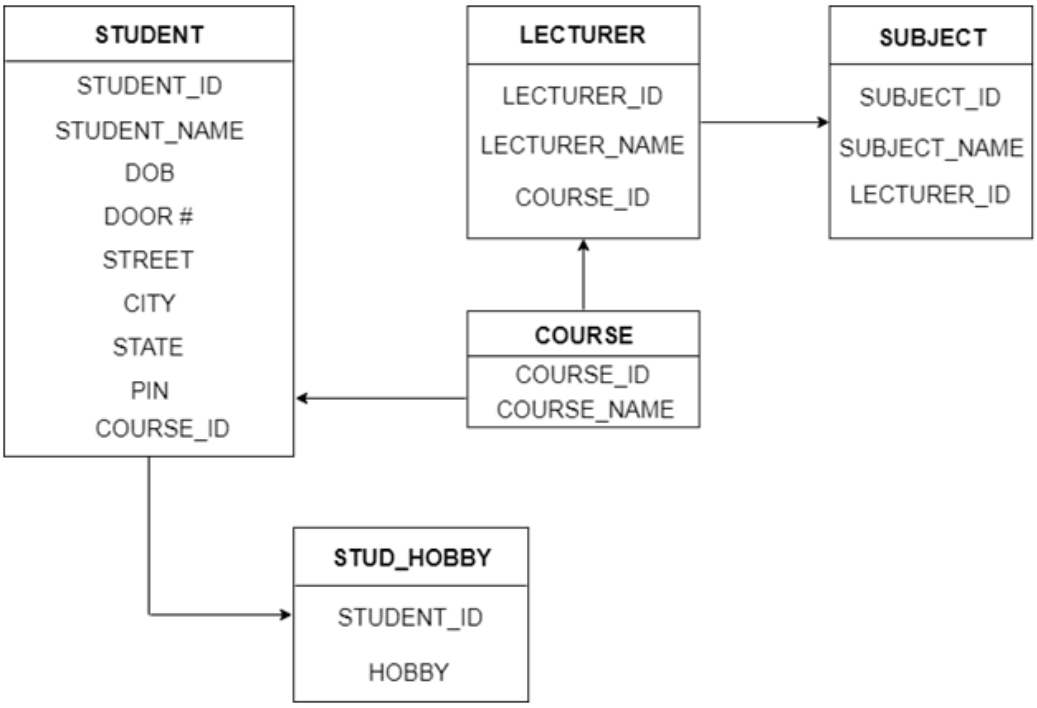


Figure: Table structure

Relational Model in DBMS

Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute.

Domain: It contains a set of atomic values that an attribute can take.

Attribute: It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom(Ai)

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.

Example: STUDENT Relation

NAME	ROLL_NO	PHONE_NO	ADDRESS	AGE
Ram	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20
Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

- In the given table, NAME, ROLL_NO, PHONE_NO, ADDRESS, and AGE are the attributes.
- The instance of schema STUDENT has 5 tuples.
- t3 = <Laxman, 33289, 8583287182, Gurugram, 20>

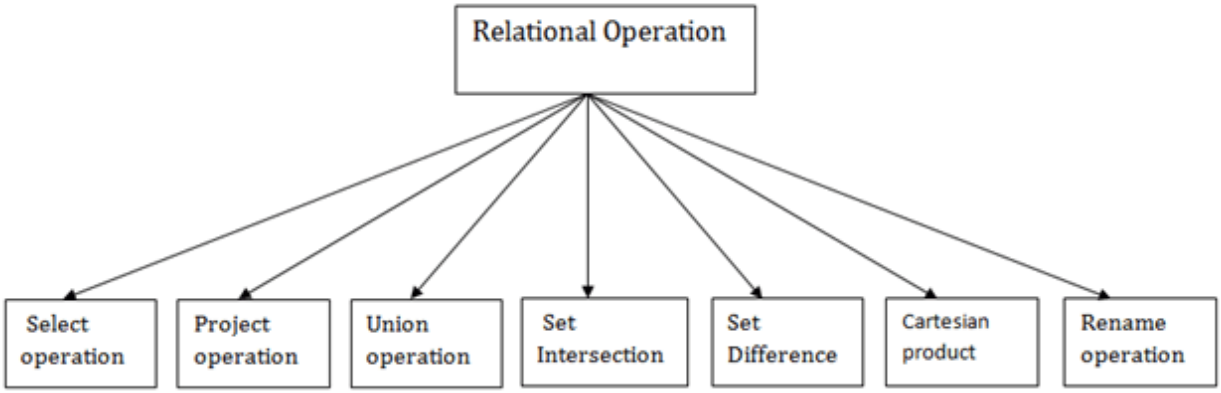
Properties of Relations

- 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

Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

Types of Relational operation



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

1. Notation: σ p(r)

Where:

σ is used for selection prediction
r is used for relation
p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000

Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

1. σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
- It is denoted by ∏.

1. Notation: ∏ A1, A2, An (r)

Where

A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye

Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

1. ∏ NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.

1. Notation: R ∪ S

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11

Williams	L-17

Input:

1. ∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

1. Notation: R ∩ S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).

1. Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

1. ∏ CUSTOMER_NAME (BORROW) − ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.

1. Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

1. EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing

2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

1. ρ(STUDENT1, STUDENT)

Join Operations:

A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ⋈.

Example:

EMPLOYEE

EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

SALARY

EMP_CODE	SALARY

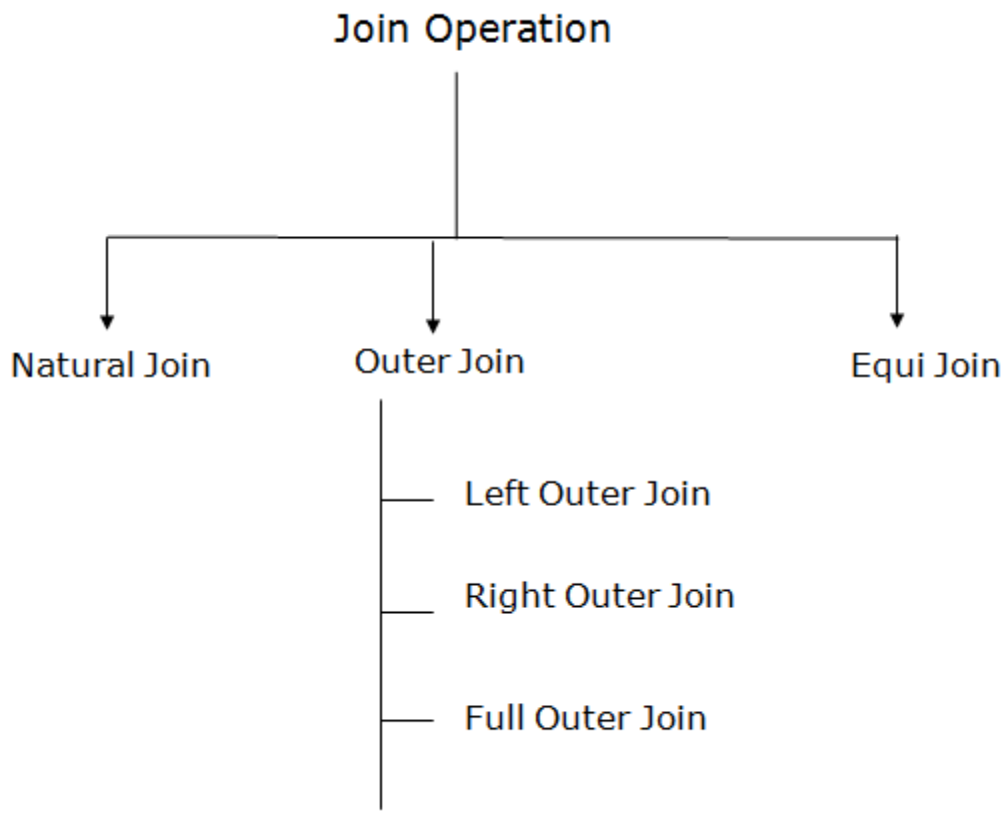
101	50000
102	30000
103	25000

1. Operation: (EMPLOYEE ⋈ SALARY)

Result:

EMP_CODE	EMP_NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000

Types of Join operations:



1. Natural Join:

- A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.
- It is denoted by ⋈.

Example: Let's use the above EMPLOYEE table and SALARY table:

Input:

1. ∏EMP_NAME, SALARY (EMPLOYEE ⋈ SALARY)

Output:

EMP_NAME	SALARY
Stephan	50000

Jack	30000
Harry	25000

2. Outer Join:

The outer join operation is an extension of the join operation. It is used to deal with missing information.

Example:

EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

Input:

1. (EMPLOYEE ⋈ FACT_WORKERS)

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru nagar	Hyderabad	TCS	50000

An outer join is basically of three types:

a. Left outer join
b. Right outer join
c. Full outer join

a. Left outer join:

- Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In the left outer join, tuples in R have no matching tuples in S.
- It is denoted by ⟕.

Example: Using the above EMPLOYEE table and FACT_WORKERS table

Input:

1. EMPLOYEE ⟕ FACT_WORKERS

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL

b. Right outer join:

- Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.

- In right outer join, tuples in S have no matching tuples in R.
- It is denoted by ⋈.

Example: Using the above EMPLOYEE table and FACT_WORKERS Relation

Input:

1. EMPLOYEE ⋈ FACT_WORKERS

Output:

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

c. Full outer join:

- Full outer join is like a left or right join except that it contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.

Example: Using the above EMPLOYEE table and FACT_WORKERS table

Input:

1. EMPLOYEE ⟗ FACT_WORKERS

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY

Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

3. Equi join:

It is also known as an inner join. It is the most common join. It is based on matched data as per the equality condition. The equi join uses the comparison operator(=).

Example:

CUSTOMER RELATION

CLASS_ID	NAME
1	John
2	Harry
3	Jackson

PRODUCT

PRODUCT_ID	CITY
1	Delhi
2	Mumbai
3	Noida

Input:

1. CUSTOMER ⋈ PRODUCT

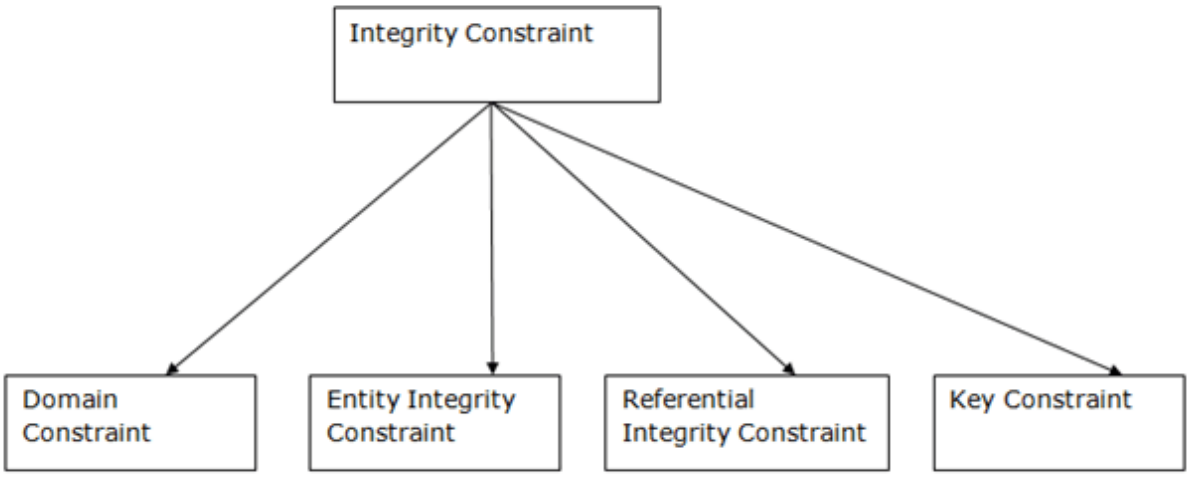
Output:

CLASS_ID	NAME	PRODUCT_ID	CITY
1	John	1	Delhi
2	Harry	2	Mumbai
3	Harry	3	Noida

Integrity Constraints

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

Types of Integrity Constraint



1. Domain constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:

(Table 1)

EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

Foreign key

Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined

Relationships

(Table 2)

Primary Key

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique