



Galgotias University

School of Computing Science and Engineering

January-2020, Semester: IV Winter: 2019-20

[Programme; B.Tech CSE] [Semester: IV] [Batch: Common to all Sections]

Course Title: Data Base Management Systems

Course Code: BCSE2011

UNIT-1

Course Material

Introduction to Data Base Management Systems

Unit I: Introduction

9 lectures

Introduction: An overview of database management system, database system vs file system, Database system concept and architecture, data model schema and instances, data independence and database language and interfaces, data definitions language, DML, Overall Database Structure. Data modeling using the Entity Relationship Model: ER model concepts, notation for ER diagram, mapping constraints, keys, Concepts of Super Key, candidate key, primary key, Generalization, aggregation, reduction of an ER diagrams to tables, extended ER model, relationship of higher degree.

Exercises:

1. Design schema and Instances.
2. Design ER diagrams for various scenarios or based on given projects.
3. Implement DDL Statements and DML statements.
4. Execute the SELECT command with different clauses.

➤ DATA BASE MANGEMENT SYSTEM:

DBMS stands for **Database Management System**.

It can break it like this DBMS = Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can define DBMS like this: DBMS is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.

DBMS also provides protection and security to the databases. It also maintains data consistency in case of multiple users.

Database Management System (DBMS) is a collection of programs which enables its users to access a database, manipulate data, reporting/representation of data.

Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks. A database management system stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

RDBMS:

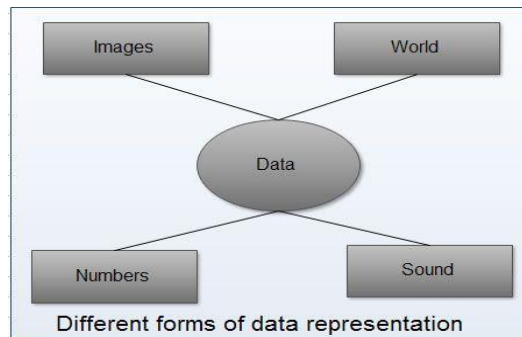
Most of today's database systems are referred to as a Relational Database Management System (RDBMS), because of their ability to store related data across multiple tables.

Here are some examples of popular DBMS used these days:

1. MySQL
2. Oracle
3. SQL Server
4. IBM DB2
5. PostgreSQL
6. Amazon SimpleDB (cloud based) etc.

➤ Data and Information

A collection of facts, such as numbers, words, text, measurements, observations or even just descriptions of things. Data can be measured, collected and reported, and analyzed. Computer data may be processed by the computer's CPU and is stored in files and folders on the computer's hard disk.

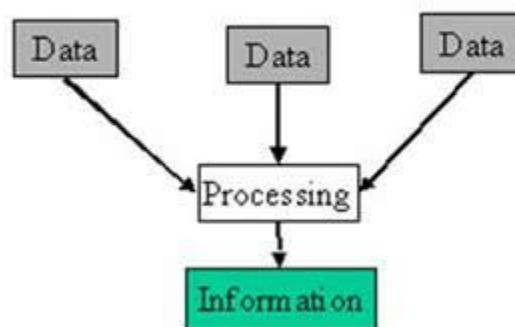


Data is raw, unorganized facts that need to be processed. **Data** can be something simple and seemingly random and useless until it is organized. When **data** is processed, organized, structured or presented in a given context so as to make it useful, it is called **information**.

Example of data: Each student's test score is one piece of data.

Example of Information: The average score of a class or of the entire school is information that can be derived from the given data.

Information is created from data



➤ Difference between Data & Information:

DATA	INFORMATION
------	-------------

Data can be represented in the form of Numbers and words which can be stored in computer's language Images, sounds, multimedia and animated data	Information is organized or classified data, which has some meaningful values for the receiver. Information is the processed data on which decisions and actions are based.
Data is considered to be raw data. It represents 'values of qualitative or quantitative variables, belonging to a set of items.' It may be in the form of numbers, letters, or a set of characters. It is often collected via measurements	Information is "knowledge communicated or received concerning a particular fact or circumstance." Information is a sequence of symbols that can be interpreted as a message. It provides knowledge or insight about a certain matter. Information can be recorded as signs, or transmitted as signals.
Qualitative Or Quantitative Variables that can be used to make ideas or conclusions	A group of data which carries news and meaning
A structure, such as tabular data, data tree, a data graph, etc.	Language, ideas, and thoughts based on the data
Not analyzed	Always analyzed
Carries no specific meaning	Carries meaning that has been assigned by interpreting data
Facts and statistics collected together for reference or analysis	Facts provided or learned about something or someone

➤ **DBMS vs. File System**

There are following differences between DBMS and File system:

DBMS	File System
DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	File system is a collection of data. In this system, the user has to write the procedures for managing the database.
DBMS gives an abstract view of data that hides the details.	File system provides the detail of the data representation and storage of data.
DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure.	File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost.
DBMS provides a good protection mechanism.	It is very difficult to protect a file under the file system.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.

DBMS takes care of Concurrent access of data using some form of locking.

In the File system, concurrent access has many problems like redirecting the file while other deleting some information or updating some information.

➤ The Need of DBMS

Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: **Storage of data** and **retrieval of data**. **Storage:** According to the principles of database systems, the data is stored in such a way that it acquires lot less space as the redundant data (duplicate data) has been removed before storage. Let's take a layman example to understand this: In a banking system, suppose a customer is having two accounts, one is saving account and another is salary account. Let's say bank stores saving account data at one place (these places are called tables we will learn them later) and salary account data at another place, in that case if the customer information such as customer name, address etc. are stored at both places then this is just a wastage of storage (redundancy/ duplication of data), to organize the data in a better way the information should be stored at one place and both the accounts should be linked to that information somehow. The same thing we achieve in DBMS. **Fast Retrieval of data:** Along with storing the data in an optimized and systematic manner, it is also important that we retrieve the data quickly when needed. Database systems ensure that the data is retrieved as quickly as possible.

➤ Characteristics of Database Management System

A database management system has following characteristics:

1. **Data stored into Tables:** Data is never directly stored into the database. Data is stored into tables, created inside the database. DBMS also allows to have relationships between tables which make the data more meaningful and connected. You can easily understand what type of data is stored where by looking at all the tables created in a database.
2. **Reduced Redundancy:** In the modern world hard drives are very cheap, but earlier when hard drives were too expensive, unnecessary repetition of data in database was a big problem. But DBMS follows **Normalization** which divides the data in such a way that repetition is minimum.
3. **Data Consistency:** On Live data, i.e. data that is being continuously updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it all by itself.
4. **Support multiple user and Concurrent Access:** DBMS allows multiple users to work on it (update, insert, delete data) at the same time and still manages to maintain the data consistency.
5. **Query Language:** DBMS provides users with a simple Query language, using which data can be easily fetched, inserted, deleted and updated in a database.
6. **Security:** The DBMS also takes care of the security of data, protecting the data from unauthorised access. In a typical DBMS, we can create user accounts with different access permissions, using which we can easily secure our data by restricting user access.
7. DBMS supports **transactions**, which allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

➤ Capabilities of Database Management System

- **Concurrent Use:** A database system allows several users to access the database concurrently. Answering different questions from different users with the same (base) data is a central aspect of an information system. Such concurrent use of data increases the economy of a system. An example for concurrent use is the travel database of a bigger travel agency. The employees of different branches can access the database concurrently and book journeys for their clients. Each travel agent sees on his interface if there are still seats available for a specific journey or if it is already fully booked.
- **Structured and Described Data:** A fundamental feature of the database approach is that the database system does not only contain the data but also the complete definition and description of these data. These descriptions are basically details about the extent, the structure, the type and the format of all data and, additionally, the relationship between the data. This kind of stored data is called metadata ("data about data").
- **Separation of Data and Applications:** As described in the feature structured data the structure of a database is described through *metadata* which is also stored in the database. Application software does not need any knowledge about the physical data storage like encoding, format, storage place, etc. It only communicates with the management system of a database (DBMS) via a standardised interface with the help of a standardised language like SQL. The access to the data and the metadata is entirely done by the DBMS. In this way all the applications can be totally separated from the data. Therefore database internal reorganisations or improvement of efficiency do not have any influence on the application software.
- **Data Integrity:** Data integrity is a byword for the quality and the reliability of the data of a database system. In a broader sense data integrity includes also the protection of the database from unauthorised access (confidentiality) and unauthorised changes. Data reflect facts of the real world database.
- **Transactions:** A transaction is a bundle of actions which are done within a database to bring it from one consistent state to a new consistent state. In between the data are inevitable inconsistent. A transaction is atomic what means that it cannot be divided up any further. Within a transaction all or none of the actions need to be carried out. Doing only a part of the actions would lead to an inconsistent database state. One example of a transaction is the transfer of an amount of money from one bank account to another. The debit of the money from one account and the credit of it to another account make together a consistent transaction. This transaction is also atomic. The debit or credit alone would both lead to an inconsistent state. After finishing the transaction (debit and credit) the changes to both accounts become persistent and the one who gave the money has now less money on his account while the receiver has now a higher balance.
- **Data Persistence:** Data persistence means that in a DBMS all data is maintained as long as it is not deleted explicitly. The life span of data needs to be determined directly or indirectly by the user and must not be dependent on system features. Additionally data once stored in a database must not be lost. Changes of a database which are done by a transaction are persistent. When a transaction is finished even a system crash cannot put the data in danger.

➤ **Advantages of DBMS**

1. Segregation of application program.
2. Minimal data duplicity or data redundancy.
3. Easy retrieval of data using the Query Language.
4. Reduced development time and maintenance need.
5. With Cloud Datacenters, we now have Database Management Systems capable of storing almost infinite data.
6. Seamless integration into the application programming languages which makes it very easier to add a database to almost any application or website.

➤ **Disadvantages of DBMS**

It's Complexity

Except MySQL, which is open source, licensed DBMSs are generally costly.

They are large in size.

➤ **Applications Of DBMS**

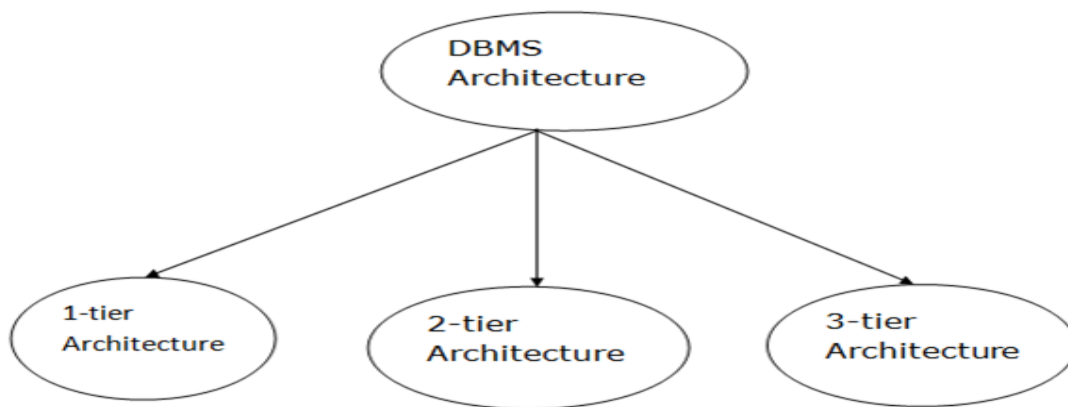
Applications where we use Database Management Systems are:

1. **Telecom:** There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
2. **Industry:** Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.
3. **Banking System:** For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
4. **Sales:** To store customer information, production information and invoice details.
5. **Airlines:** To travel though airlines, we make early reservations, this reservation information along with flight schedule is stored in database.
6. **Education sector:** Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.
7. **Online shopping:** You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

➤ **DBMS Architecture**

1. 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.
2. The client/server architecture consists of many PCs and a workstation which are connected via the network.
3. 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

1. In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
2. Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
3. 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

1. 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.
2. The user interfaces and application programs are run on the client-side.
3. The server side is responsible to provide the functionalities like: query processing and transaction management.
4. To communicate with the DBMS, client-side application establishes a connection with the server side.

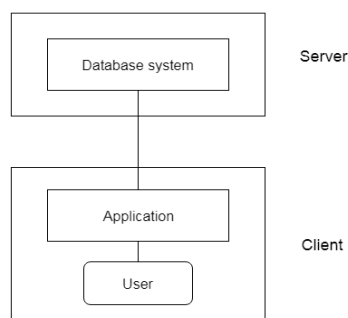


Fig: 2-tier Architecture

3-Tier Architecture

1. The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
2. The application on the client-end interacts with an application server which further communicates with the database system.
3. 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.
4. The 3-Tier architecture is used in case of large web application.

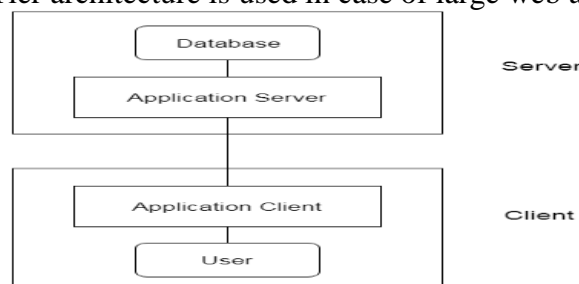


Figure: Three tier architecture

➤ Three schema Architecture

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

The three-schema architecture is as follows:

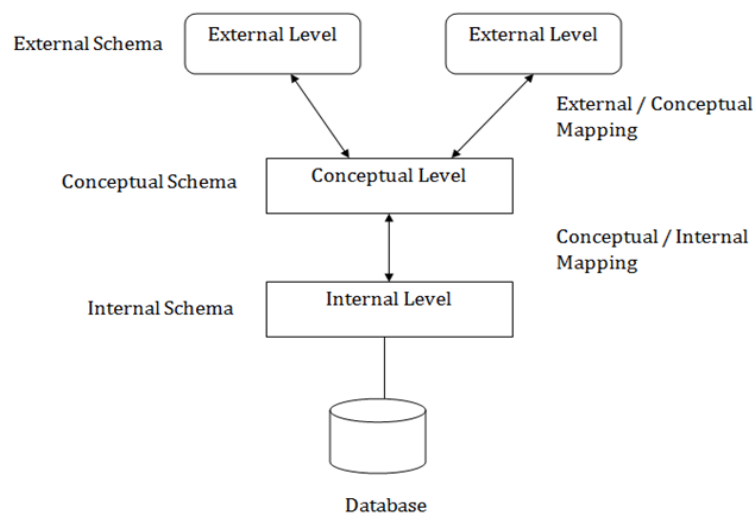


Figure: Three Schema Architecture

In the above diagram:

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

1. Internal Level

1. The internal level has an internal schema which describes the physical storage structure of the database.

2. The internal schema is also known as a physical schema.
3. It uses the physical data model. It is used to define that how the data will be stored in a block.
4. The physical level is used to describe complex low-level data structures in detail.

2. Conceptual Level

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

3. External Level

1. 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.
2. An external schema is also known as view schema.
3. Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
4. The view schema describes the end user interaction with database systems.

➤ Data model Schema and Instance

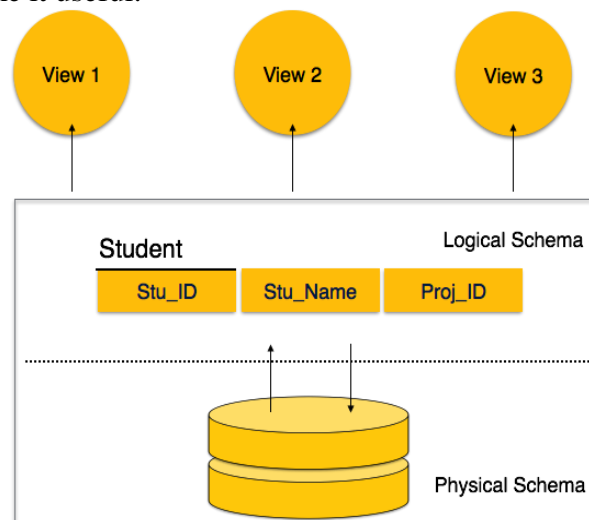
1. The data which is stored in the database at a particular moment of time is called an instance of the database.
2. The overall design of a database is called schema.
3. A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
4. A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
5. A database schema can be represented by using the visual diagram. That diagram shows the database objects and relationship with each other.
6. 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.

➤ Database Schema

- A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.
- A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.



A database schema can be divided broadly into two categories –

- **Physical Database Schema** – This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema** – This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Instance

- It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.
- A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

Data Independence

1. Data independence can be explained using the three-schema architecture.

2. 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.

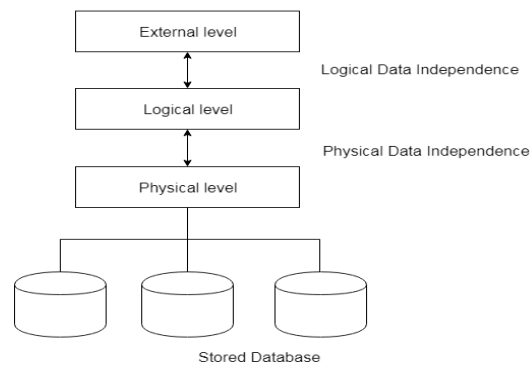


Fig: Data Independence

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

➤ **USERS OF DBMS**

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

- **Administrators:** Administrators maintain the DBMS and are responsible for administrating the database. They are responsible to look after its usage and by whom it should be used. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.
- **Designers:** Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in what format. They identify and design the whole set of entities, relations, constraints, and views. End Users: End users are those who actually reap the benefits of having a DBMS.
- **End User:** End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

➤ **DBMS - Data Models**

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

The very first data model could be flat data-models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies.

➤ DBMS Database Models

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

Hierarchical Model

This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.

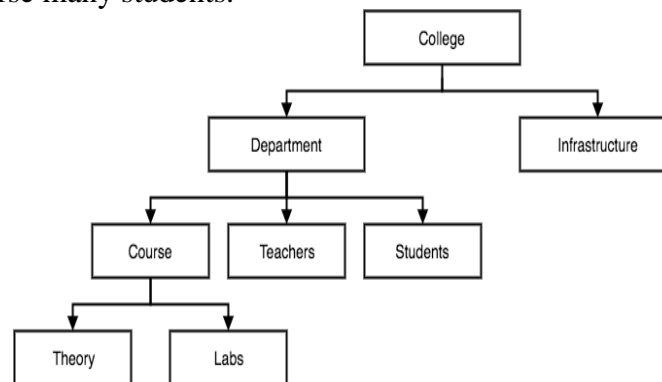


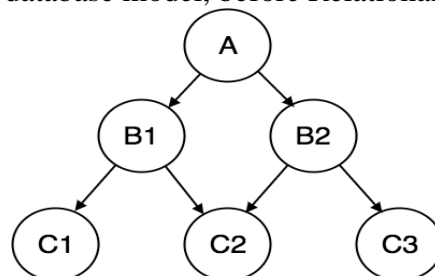
Figure : Example of Hierarchical Model

Network Model

This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before Relational Model was introduced.



Entity-relationship Model

- In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.
- Different entities are related using relationships.
- E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.
- This model is good to design a database, which can then be turned into tables in relational model(explained below).
- Let's take an example, If we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.
- Relationships can also be of different types. To learn about [E-R Diagrams](#) in details, click on the link.

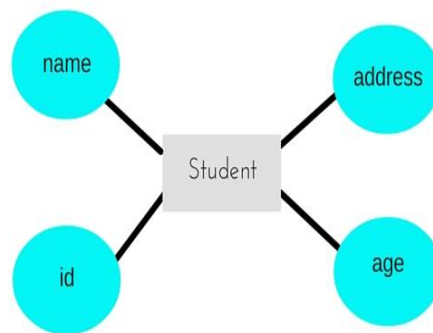


Figure: Entity Relational Model

Relational Model

- In this model, data is organized in two-dimensional **tables** and the relationship is maintained by storing a common field.
- This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, in fact, we can say the only database model used around the world.
- The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.
- Hence, tables are also known as **relations** in relational model.
- In the coming tutorials we will learn how to design tables, normalize them to reduce data redundancy and how to use Structured Query language to access data from tables.

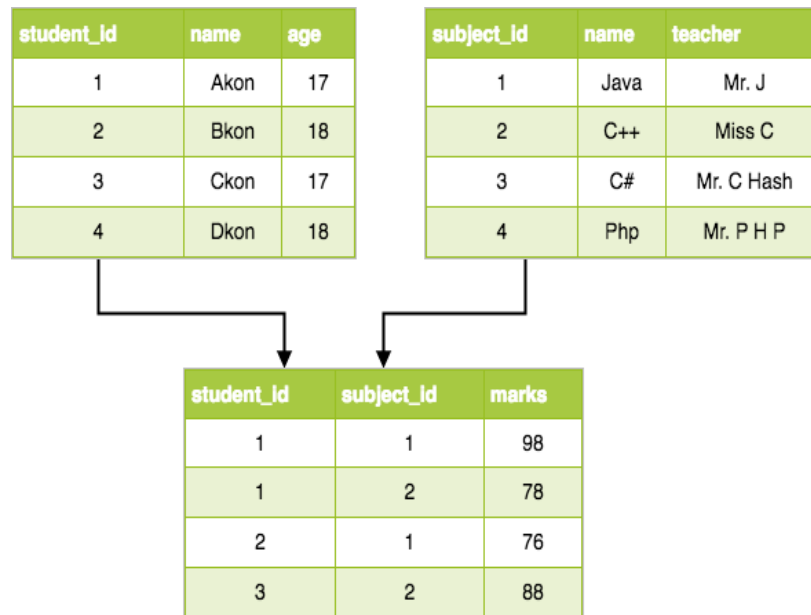


Figure : Example of Relational Model

Entity-Relationship Model

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

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

ER Model is based on –

- **Entities** and their *attributes*.
- **Relationships** among entities.

These concepts are explained below.

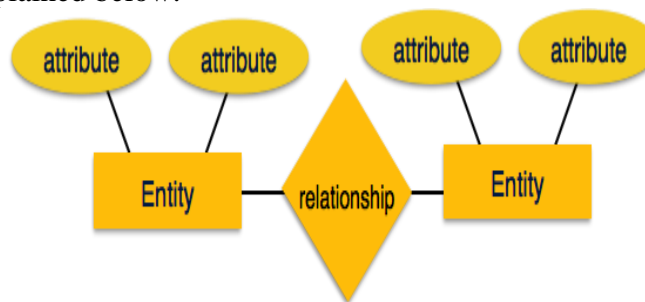


Figure : Example of Entity Relationship Model

- **Entity** – An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.
- **Relationship** – The logical association among entities is called **relationship**. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

Mapping cardinalities –

- one to one

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

Relational Model

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

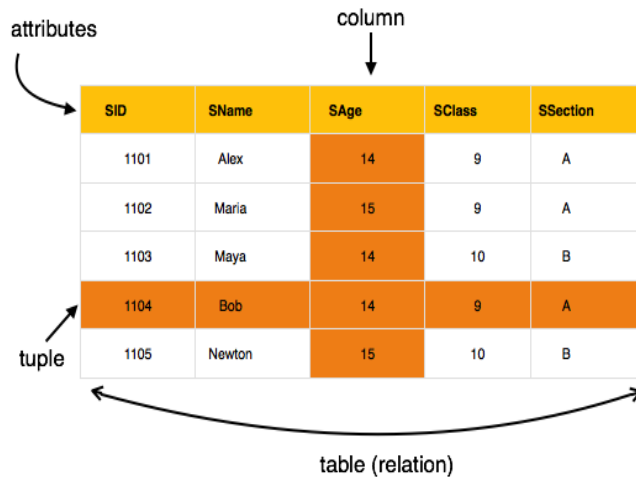


Figure: Example of Relational Model

The main highlights of this model are –

- Data is stored in tables called **relations**.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

➤ Basic Concepts of ER Model in DBMS

As we described in the tutorial Database models, Entity-relationship model is a model used for design and representation of relationships between data.

The main data objects are termed as Entities, with their details defined as attributes, some of these attributes are important and are used to identity the entity, and different entities are related using relationships.

In short, to understand about the ER Model, we must understand about:

- Entity and Entity Set
- What are Attributes? And Types of Attributes.
- Keys
- Relationships

Let's take an example to explain everything. For a **School Management Software**, we will have to store **Student** information, **Teacher** information, **Classes**, **Subjects** taught in each class etc.

ER Model: Entity and Entity Set

Considering the above example, **Student** is an entity, **Teacher** is an entity, similarly, **Class**, **Subject** etc are also entities.

An Entity is generally a real-world object which has characteristics and holds relationships in a DBMS.

If a Student is an Entity, then the complete dataset of all the students will be the **Entity Set**

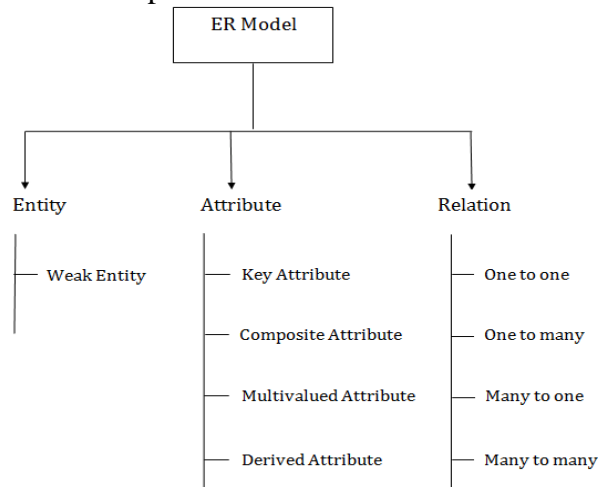


Figure : ER Model

ER Model: Attributes

If a Student is an Entity, then student's **roll no.**, student's **name**, student's **age**, student's **gender** etc will be its attributes.

An attribute can be of many types, here are different types of attributes defined in ER database model:

- **Simple attribute:** The attributes with values that are atomic and cannot be broken down further are simple attributes. For example, student's **age**.
- **Composite attribute:** A composite attribute is made up of more than one simple attribute. For example, student's **address** will contain, **house no.**, **street name**, **pincode** etc.
- **Derived attribute:** These are the attributes which are not present in the whole database management system, but are derived using other attributes. For example, *average age of students in a class*.
- **Single-valued attribute:** As the name suggests, they have a single value.
- **Multi-valued attribute:** And, they can have multiple values.

ER Model: Keys

If the attribute **roll no.** can uniquely identify a student entity, amongst all the students, then the attribute **roll no.** will be said to be a key.

Following are the types of Keys:

- Super Key
- Candidate Key
- Primary Key

ER Model: Relationships

When an Entity is related to another Entity, they are said to have a relationship. For example, A **Class** Entity is related to **Student** entity, because students study in classes, hence this is a relationship.

Depending upon the number of entities involved, a **degree** is assigned to relationships.

For example, if 2 entities are involved, it is said to be **Binary relationship**, if 3 entities are involved, it is said to be **Ternary relationship**, and so on.

In the next tutorial, we will learn how to create ER diagrams and design databases using ER diagrams.

Working with ER Diagrams

ER Diagram is a visual representation of data that describes how data is related to each other. In ER Model, we disintegrate data into entities, attributes and setup relationships between entities, all this can be represented visually using the ER diagram.

For example, in the below diagram, anyone can see and understand what the diagram wants to convey: *Developer develops a website, whereas a Visitor visits a website.*

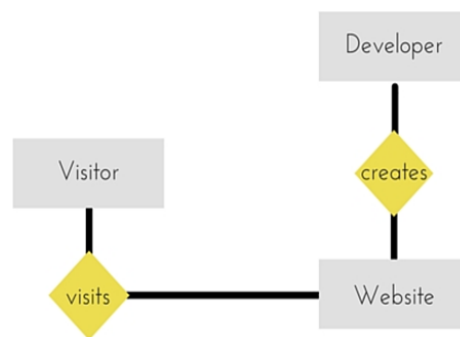


Figure : Example of ER Diagram

Components of ER Diagram

Entity, Attributes, Relationships etc form the components of ER Diagram and there are defined symbols and shapes to represent each one of them.

Let's see how we can represent these in our ER Diagram.

Entity

Simple rectangular box represents an Entity.



Figure : Example of Entities

Relationships between Entities - Weak and Strong

Rhombus is used to setup relationships between two or more entities.

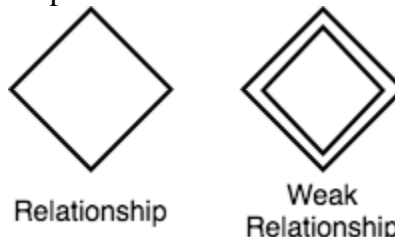


Figure : Relationship

Attributes for any Entity

Ellipse is used to represent attributes of any entity. It is connected to the entity.

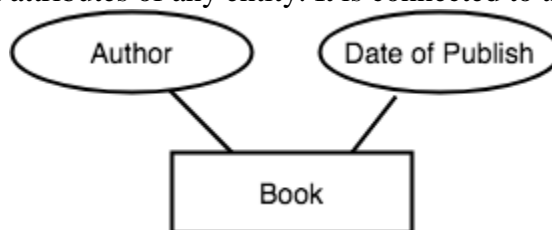


Figure : Attributes for Entity

Weak Entity

A weak Entity is represented using double rectangular boxes. It is generally connected to another entity.



Figure : Weak Entity

Key Attribute for any Entity

To represent a Key attribute, the attribute name inside the Ellipse is underlined.

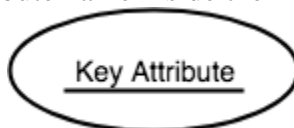


Figure : Key Attribute

Derived Attribute for any Entity

Derived attributes are those which are derived based on other attributes, for example, age can be derived from date of birth.

To represent a derived attribute, another dotted ellipse is created inside the main ellipse.



Figure : Derived Attribute

Multivalued Attribute for any Entity

Double Ellipse, one inside another, represents the attribute which can have multiple values.



Figure : Multivalued Attribute

Composite Attribute for any Entity

A composite attribute is the attribute, which also has attributes.

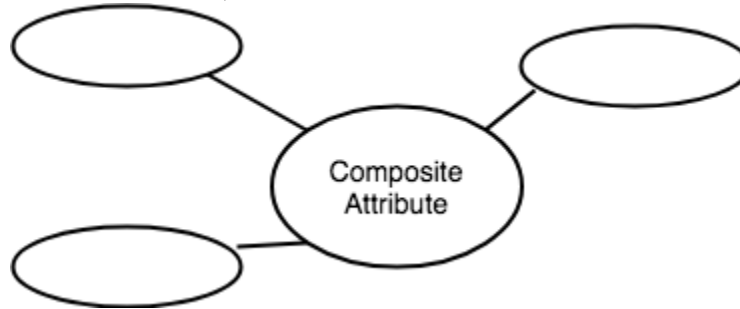
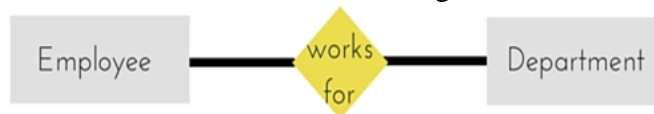


Figure: Composite Attribute

ER Diagram: Entity

An **Entity** can be any object, place, person or class. In ER Diagram, an **entity** is represented using rectangles. Consider an example of an Organisation- Employee, Manager, Department, Product and many more can be taken as entities in an Organisation.



The yellow rhombus in between represents a relationship.

ER Diagram: Weak Entity

Weak entity is an entity that depends on another entity. Weak entity doesn't have any key attribute of its own. Double rectangle is used to represent a weak entity.

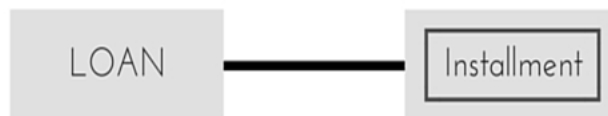


Figure : Weak Entity

ER Diagram: Attribute

An **Attribute** describes a property or characteristic of an entity. For example, **Name**, **Age**, **Address** etc can be attributes of a **Student**. An attribute is represented using eclipse.

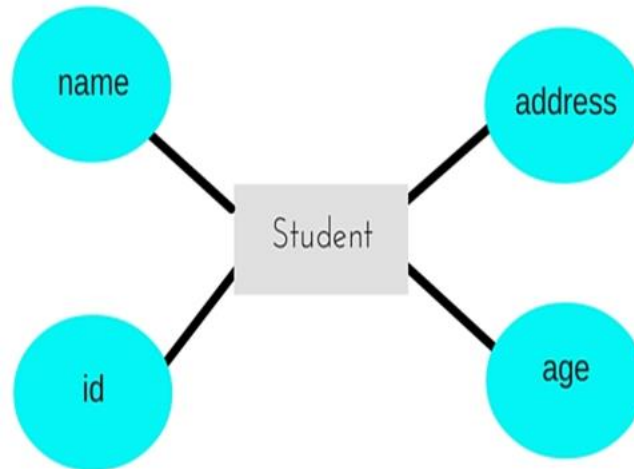


Figure: ER Diagram

ER Diagram: Key Attribute

Key attribute represents the main characteristic of an Entity. It is used to represent a Primary key. Ellipse with the text underlined, represents Key Attribute.

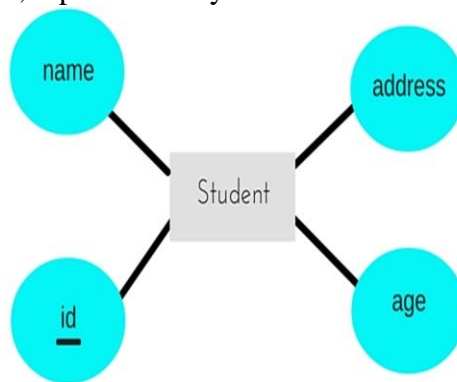


Figure : ER Diagram

ER Diagram: Composite Attribute

An attribute can also have their own attributes. These attributes are known as **Composite** attributes.

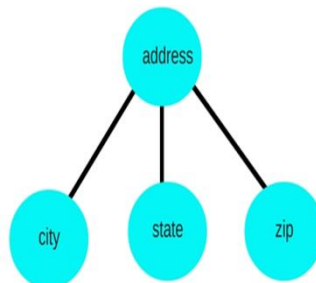
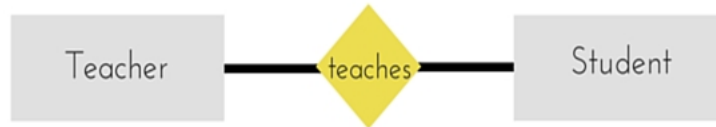


Figure :Composite Attribute

ER Diagram: Relationship

A Relationship describes relation between **entities**. Relationship is represented using diamonds or rhombus.



There are three types of relationship that exist between Entities.

- Binary Relationship
- Recursive Relationship
- Ternary Relationship

ER Diagram: Binary Relationship

Binary Relationship means relation between two Entities. This is further divided into three types.

One to One Relationship

This type of relationship is rarely seen in real world.

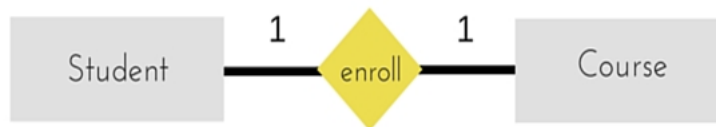


Figure : one to one Relation ship

The above example describes that one student can enroll only for one course and a course will also have only one Student. This is not what you will usually see in real-world relationships.

One to Many Relationship

The below example showcases this relationship, which means that 1 student can opt for many courses, but a course can only have 1 student. Sounds weird! This is how it is.

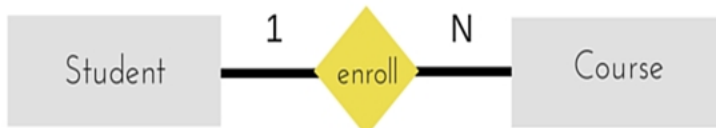


Figure : One to Many Relationship

Many to One Relationship

It reflects business rule that many entities can be associated with just one entity. For example, Student enrolls for only one Course but a Course can have many Students.

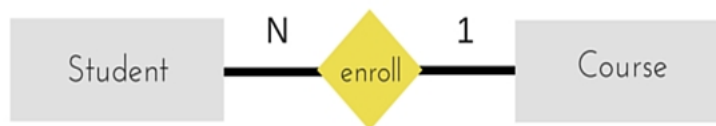


Figure : Many to One Relationship

Many to Many Relationship

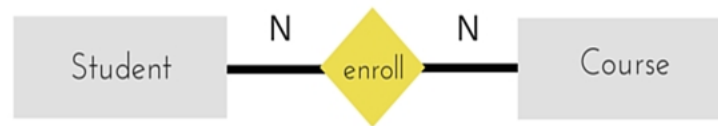
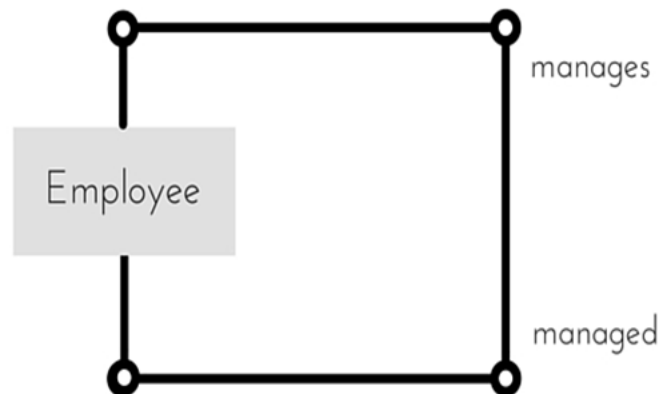


Figure : Many to Many Relationship

The above diagram represents that one student can enroll for more than one courses. And a course can have more than 1 student enrolled in it.

ER Diagram: Recursive Relationship

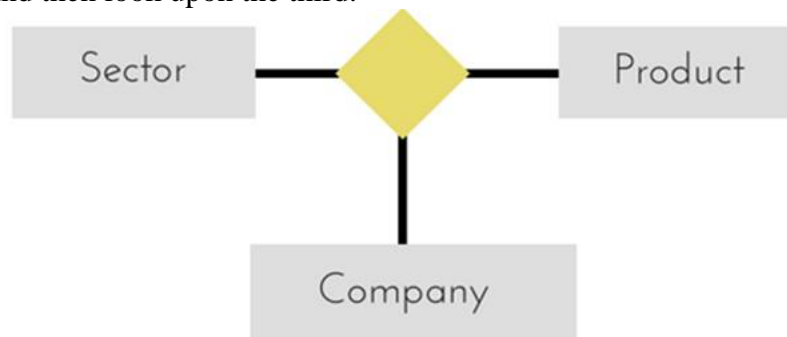
When an Entity is related with itself it is known as **Recursive Relationship**.



ER Diagram: Ternary Relationship

Relationship of degree three is called Ternary relationship.

A Ternary relationship involves three entities. In such relationships we always consider two entities together and then look upon the third.



- The above relationship involves 3 entities.
- Company operates in Sector, producing some Products.

Figure: Ternary Relation

- For example, in the diagram above, we have three related entities,
- **Company, Product** and **Sector**. To understand the relationship better or to define rules around the model, we should relate two entities and then derive the third one.

- A **Company** produces many **Products**/ each product is produced by exactly one company.
- A **Company** operates in only one **Sector** / each sector has many companies operating in it.

Considering the above two rules or relationships, we see that although the complete relationship involves three entities, but we are looking at two entities at a time.

➤ **The Enhanced ER Model**

As the complexity of data increased in the late 1980s, it became more and more difficult to use the traditional ER Model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.

Hence, as part of the **Enhanced ER Model**, along with other improvements, three new concepts were added to the existing ER Model, they were:

1. Generalization
2. Specialization
3. Aggregation

Let's understand what they are, and why were they added to the existing ER Model.

Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entities to make further higher level entity.

It's more like Superclass and Subclass system, but the only difference is the approach, which is bottom-up. Hence, entities are combined to form a more generalised entity, in other words, sub-classes are combined to form a super-class.

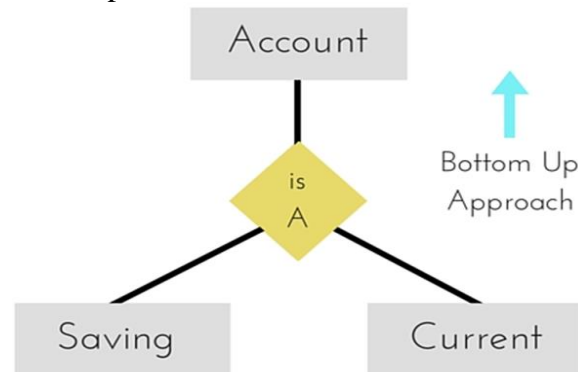


Figure : Generalization

For example, **Saving** and **Current** account types entities can be generalised and an entity with name **Account** can be created, which covers both.

Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entity. In specialization, a higher level entity may not have any lower-level entity sets, it's possible.

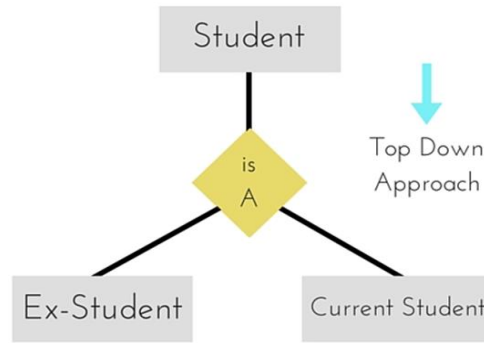


Figure: Specialization

Aggregation

Aggregation is a process when relation between two entities is treated as a **single entity**.

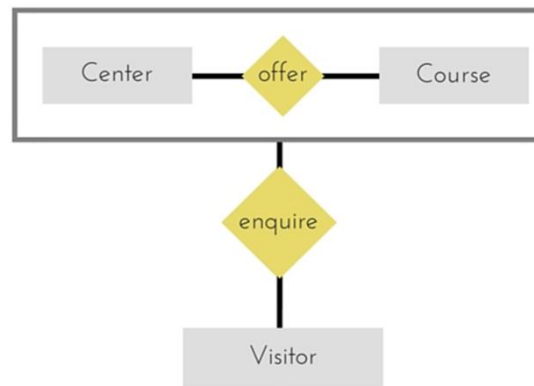


Figure: Aggregation

In the diagram above, the relationship between **Center** and **Course** together, is acting as an Entity, which is in relationship with another entity **Visitor**. Now in real world, if a Visitor or a Student visits a Coaching Center, he/she will never enquire about the center only or just about the course, rather he/she will ask enquire about both.

➤ Introduction to Database Keys

Keys are very important part of Relational database model. They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.

A Key can be a single attribute or a group of attributes, where the combination may act as a key.

The need of Key

In real world applications, number of tables required for storing the data is huge, and the different tables are related to each other as well.

Also, tables store a lot of data in them. Tables generally extends to thousands of records stored in them, unsorted and unorganised.

Now to fetch any particular record from such dataset, you will have to apply some conditions, but what if there is duplicate data present and every time you try to fetch some data by applying certain condition, you get the wrong data. How many trials before you get the right data?

To avoid all this, **Keys** are defined to easily identify any row of data in a table.

Let's try to understand about all the keys using a simple example.

Student_id	Name	Phone	Age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

Let's take a simple **Student** table, with fields **student_id**, **name**, **phone** and **age**.

Super Key

Super Key is defined as a set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.

In the table defined above super key would include **student_id**, (**student_id**, **name**), **phone** etc.

Confused? The first one is pretty simple as **student_id** is unique for every row of data, hence it can be used to identify each row uniquely.

Next comes, (**student_id**, **name**), now name of two students can be same, but their **student_id** can't be same hence this combination can also be a key.

Similarly, phone number for every student will be unique, hence again, **phone** can also be a key.

So they all are super keys.

Candidate Key

Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table. There can be more than one candidate key.


In our example, **student_id** and **phone** both are candidate keys for table **Student**.

- A candidate key can never be NULL or empty. And its value should be unique.
- There can be more than one candidate keys for a table.
- A candidate key can be a combination of more than one columns(attributes).

Primary Key

Primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.

Primary Key for this table



student_id	name	age	phone

Figure : PrimaryKey

For the table **Student** we can make the **student_id** column as the primary key.

Composite Key

Key that consists of two or more attributes that uniquely identify any record in a table is called **Composite key**. But the attributes which together form the **Composite key** are not a key independently or individually.

Composite Key
↑

student_id	subject_id	marks	exam_name

Score Table - To save scores of the student for various subjects.

Figure : Composite Key

In the above picture we have a **Score** table which stores the marks scored by a student in a particular subject.

In this table **student_id** and **subject_id** together will form the primary key, hence it is a composite key.

Secondary or Alternative key

The candidate key which are not selected as primary key are known as secondary keys or alternative keys.

Non-key Attributes

Non-key attributes are the attributes or fields of a table, other than **candidate key** attributes/fields in a table.

Non-prime Attributes

Non-prime Attributes are attributes other than **Primary Key attribute(s)**.

Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert etc. to carry out the required tasks.

These SQL commands are mainly categorized into four categories as:

DDL – Data Definition Language

DQL – Data Query Language

DML – Data Manipulation Language

DCL – Data Control Language

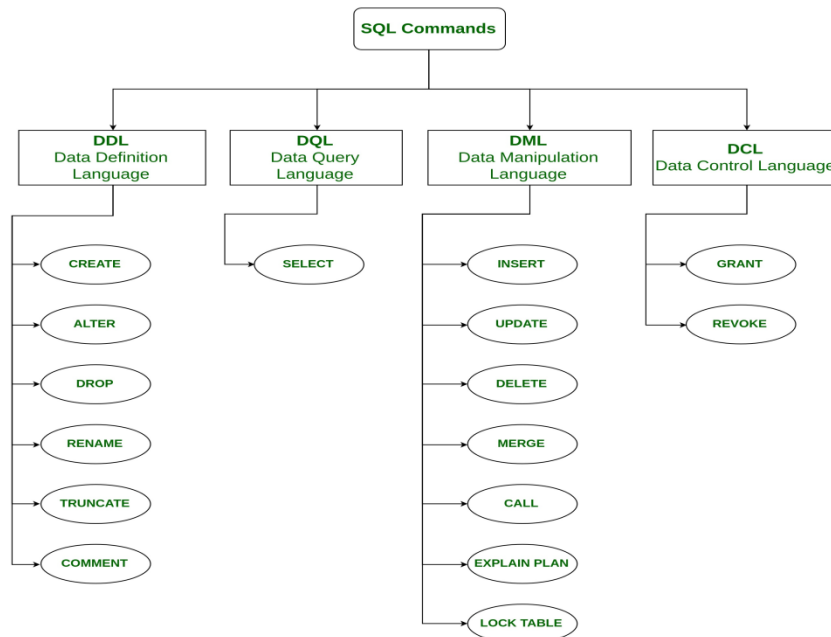


Figure: Types of SQL Commands

Though many resources claim there to be another category of SQL clauses TCL – Transaction Control Language. So we will see in detail about TCL as well

DDL(Data Definition Language) : DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

Examples of DDL commands:

CREATE – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).

DROP – is used to delete objects from the database.

ALTER-is used to alter the structure of the database.

TRUNCATE–is used to remove all records from a table, including all spaces allocated for the records are removed.

COMMENT –is used to add comments to the data dictionary.

RENAME –is used to rename an object existing in the database.

DQL (Data Query Language) :

DML statements are used for performing queries on the data within schema objects. The purpose of DQL Command is to get some schema relation based on the query passed to it.

Example of **DQL**:

SELECT – is used to retrieve data from the a database.

DML(Data Manipulation Language) : The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

Examples of DML:

INSERT – is used to insert data into a table.

UPDATE – is used to update existing data within a table.

DELETE – is used to delete records from a database table.

DCL(Data Control Language) : DCL includes commands such as **GRANT** and **REVOKE** which mainly deals with the rights, permissions and other controls of the database system.

Examples of DCL commands:

GRANT-gives user's access privileges to database.

REVOKE-withdraw user's access privileges given by using the **GRANT** command.

TCL(transaction Control Language) : TCL commands deals with the transaction within the database.

Examples of TCL commands:

COMMIT– commits a Transaction.

ROLLBACK– rollbacks a transaction in case of any error occurs.

SAVEPOINT–sets a savepoint within a transaction.

SET TRANSACTION–specify characteristics for the transaction

Title	Data Definition Language
Objective	Study of Data Definition language commands. - Create table, Alter Table,Drop Table, Rename Table.

Pre-requisite	Knowledge of Basic Database
Algorithm /Theory	<p>The SQL DDL allows specification of not only a set of relations but also information about each relation, including-</p> <ul style="list-style-type: none"> • Schema for each relation • The domain of values associated with each attribute. • The integrity constraints. • The set of indices to be maintained for each relation. • The security and authorization information for each relation. <p>The physical storage structure of each relation on disk.</p>
Syntax	<p><u>CREATE TABLE</u></p> <p>CREATE TABLE TABLENAME(COLUMN_NAME1 DATA_TYPE1(SIZE1),..... COLUMN_NAMEN DATA_TYPEN(SIZEN));</p> <p><u>ALTER TABLE</u></p> <p>ALTER TABLE <i>table_name</i> ADD <i>column_name datatype</i>;</p> <p>ALTER TABLE <i>table_name</i> MODIFY <i>column_name datatype</i>;</p> <p>ALTER TABLE <i>table_name</i> DROP COLUMN <i>column_name</i>;</p> <p>ALTER TABLE <i>table_name</i> RENAME COLUMN <i>OLDcolumn_name TO NEWcolumn_name</i>;</p> <p><u>DROP TABLE</u></p> <p>DROP TABLE <i>table_name</i>;</p> <p><u>RENAME TABLE</u></p> <p>RENAME <i>OLDtable_name TO NEWtable_name</i>;</p> <p><u>TRUNCATE</u></p> <p>TRUNCATE TABLE <i>table_name</i>;</p>

Title	Data Manipulation Language Statements.
Objective	Study of Data Manipulation Statements.

Pre-requisite	Knowledge of <ul style="list-style-type: none"> ORACLE Queries
Algorithm /Theory	<p>Data Manipulation Language (DML) statements are used for managing data in database. DML commands are not auto-committed. It means changes made by DML command are not permanent to database, it can be rolled back.</p> <p>DML statements are used for managing data within schema objects. Some examples:</p> <ul style="list-style-type: none"> SELECT - retrieve data from the a database INSERT - insert data into a table UPDATE - updates existing data within a table DELETE - deletes all records from a table, the space for the records remain
Syntax	<pre> INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...); UPDATE table_name SET column1 = value1, column2 = value2, ... WHERE condition; DELETE FROM table_name WHERE condition; SELECT column1, column2, ...FROM table_name; </pre>

Title	SELECT Command
Objective	Study of SELECT command with different clauses.
Pre-requisite	Knowledge of <ul style="list-style-type: none"> ORACLE
Algorithm /Theory	<p>SQL SELECT Statement</p> <p>The most commonly used SQL command is SELECT statement. SQL SELECT statement is used to query or retrieve data from a table in the database. A query may retrieve information from specified columns or from all of the columns in the table. To create a simple SQL SELECT Statement, you must specify the column(s) name and the table name. The whole query is called SQL SELECT Statement.</p>
Syntax	<p>Syntax of SQL SELECT Statement:</p> <pre> SELECT column_list FROM table-name [WHERE Clause] </pre>

	<p>[GROUP BY clause]</p> <p>[HAVING clause]</p> <p>[ORDER BY clause];</p>
--	---

Subject Matter for students to prepare self-notes, taken from different books/Journals/Internet

1. https://www.tutorialspoint.com/dbms/dbms_tutorial.pdf
2. https://www.tutorialspoint.com/dbms/dbms_overview.htm
3. <https://www.guru99.com/dbms-tutorial.html>
4. <https://www.javatpoint.com/dbms-tutorial>
5. <https://www.studytonight.com/dbms/overview-of-dbms.php>