Semester-4

(Database Management System)

(According to Purvanchal University Syllabus)

Unit – 1

Introduction

- A database is one of the important components for many applications and is used for storing a series of data in a single set. In other words, it is a group/package of information that is put in order so that it can be easily accessed, manage and update.

An overview of database management system–

- 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.

Characteristics

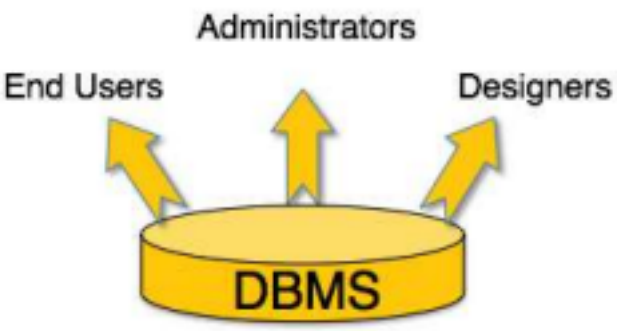
Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics −

- **Real-world entity −** A modern DBMS is more realistic and uses real world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.
- **Relation-based tables −** DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
- **Isolation of data and application −** A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.
- **Less redundancy −** DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.

- **Consistency −** Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.
- **Query Language −** DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
- **ACID Properties −** DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi
transactional environments and in case of failure.
- **Multiuser and Concurrent Access −** DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.
- **Multiple views −** DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

Users

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



Administrators – A database administrator (DBA) is a person or group in charge of implementing **DBMS** in an organization. The DBA job requires a high degree of technical expertise. DBA consists of a team of people rather than just one person.

The primary role of Database administrator is as follows −

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 users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

Database system vs File system–

Basis	DBMS Approach	File System Approach
Meaning	DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	The file system is a collection of data. In this system, the user has to write the procedures for managing the database.
Sharing of data	Due to the centralized approach, data sharing is easy.	Data is distributed in many files, and it may be of different formats, so it isn't easy to share data.

Data Abstraction	DBMS gives an abstract view of data that hides the details.	The file system provides the detail of the data representation and storage of data.
Security and Protection	DBMS provides a good protection mechanism.	It isn't easy to protect a file under the file system.
Recovery Mechanism	DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from system failure.	The 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 be lost.

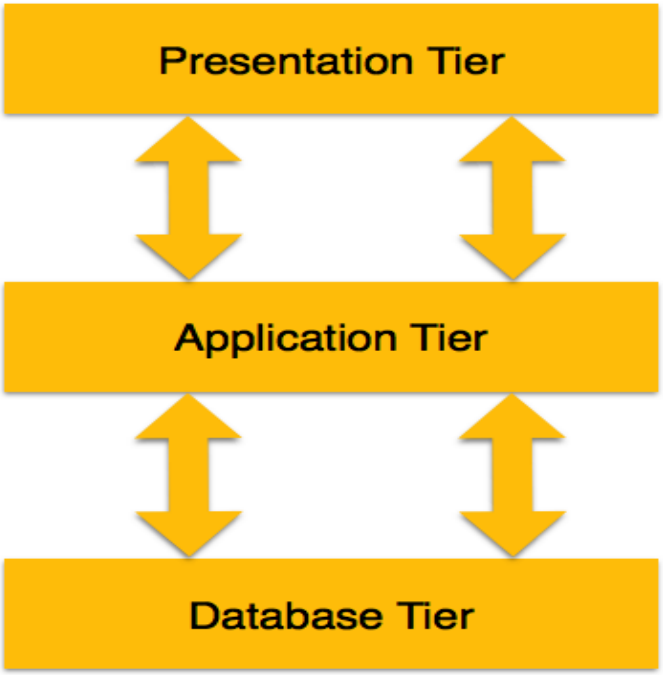
Database system concepts and Architecture–

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.

3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database.

It is the most widely used architecture to design a DBMS.



Data models schema and instances–

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 s same plane. Earlier data models were not so scientific,

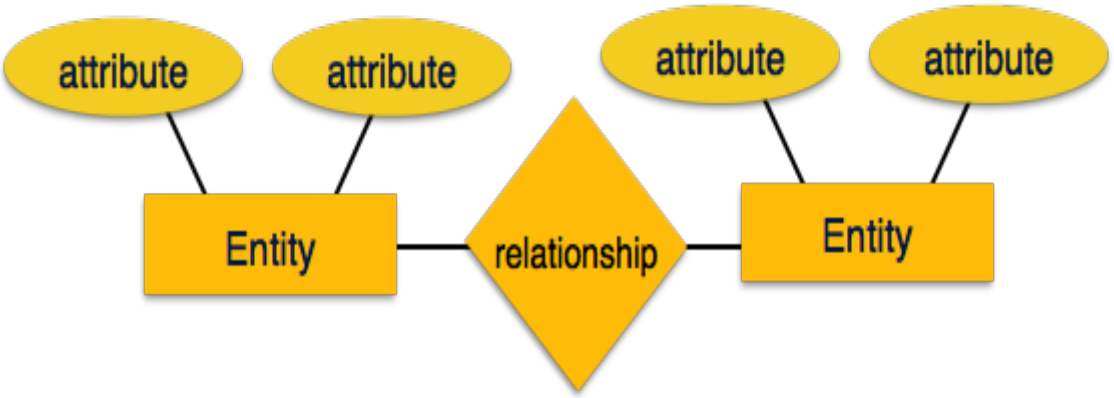
hence they were prone to introduce lots of duplication and update anomalies.

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 –



Entity –
An entity in an ER Model is a real-world entity having properties called

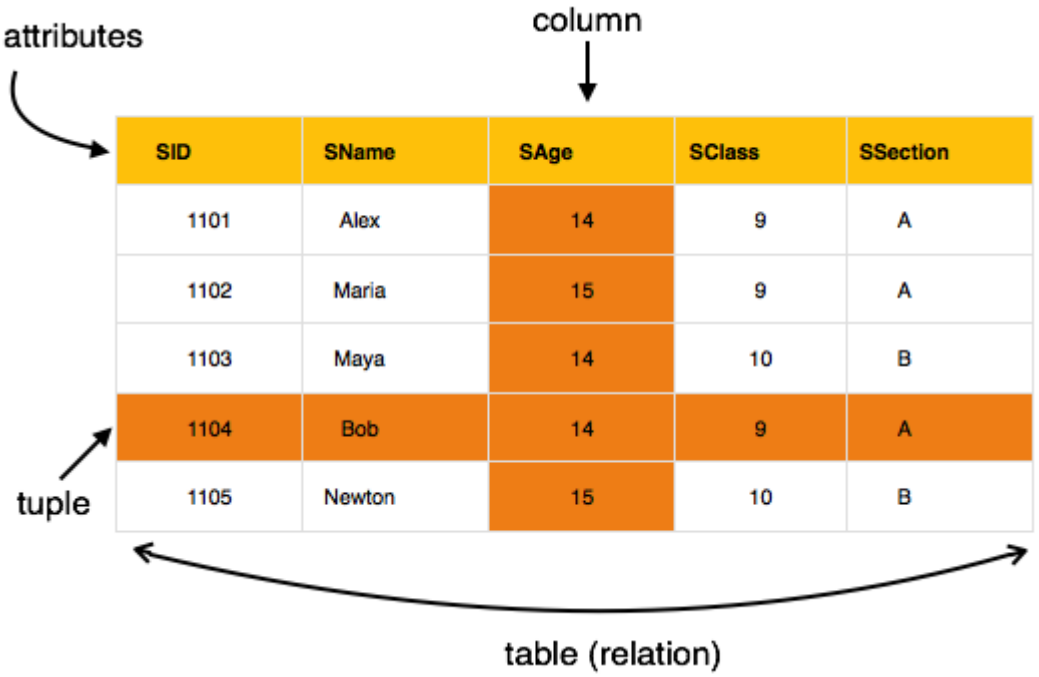
attributes-
Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.

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-

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



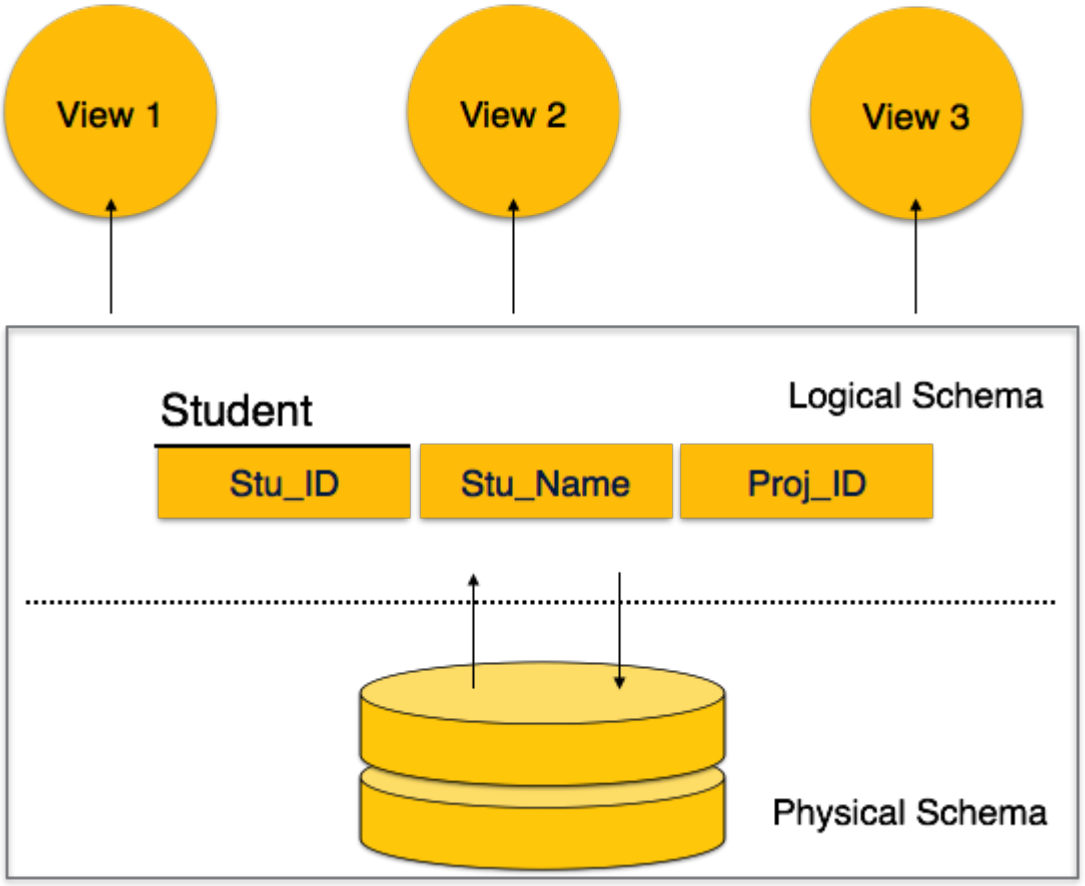
The main highlights of this model are –

- ❖ Data is stored in tables 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.

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
how the relations among them are associated.

- 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
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 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.

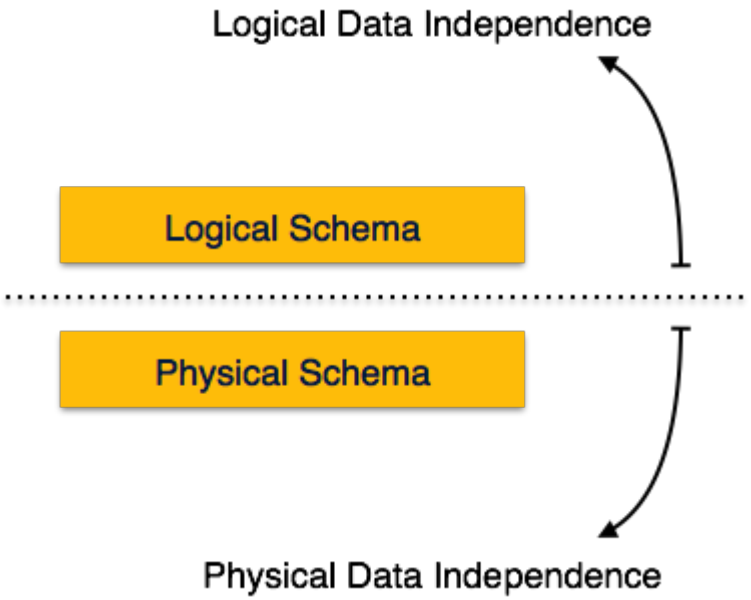
Database Instance

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database.to change with time.

Data independence and database language and interfaces–

Data Independence

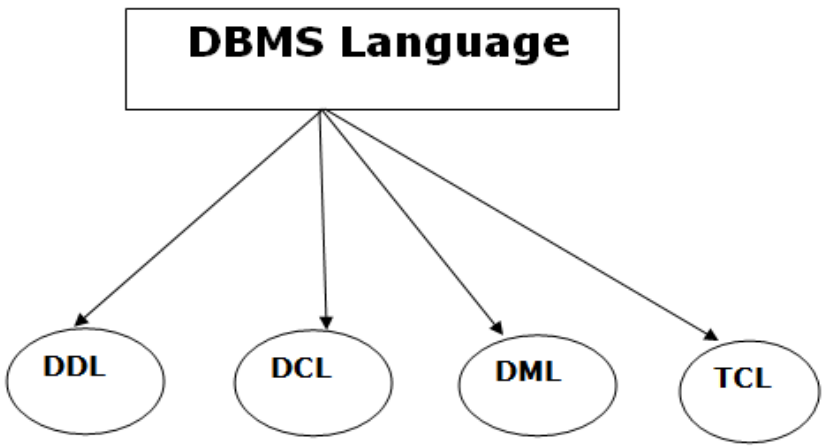
- When we are change in application data is not change is known as data independence.



Database Language

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database.

Types of Database Language



1. Data Definition Language

- **DDL** stands for Data Definition Language. It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of the number of tables and schemas, their names, indexes, columns table, constraints, etc.

Here are some tasks that come under DDL:

- **Create**: It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

2. Data Manipulation Language

DML stands for Data Manipulation Language. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge**: It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

3. Data Control Language

- DCL stands for Data Control Language. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.

  (But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user. There are the

following operations which have the authorization of Revoke: CONNECT,

INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT. 4. Transaction

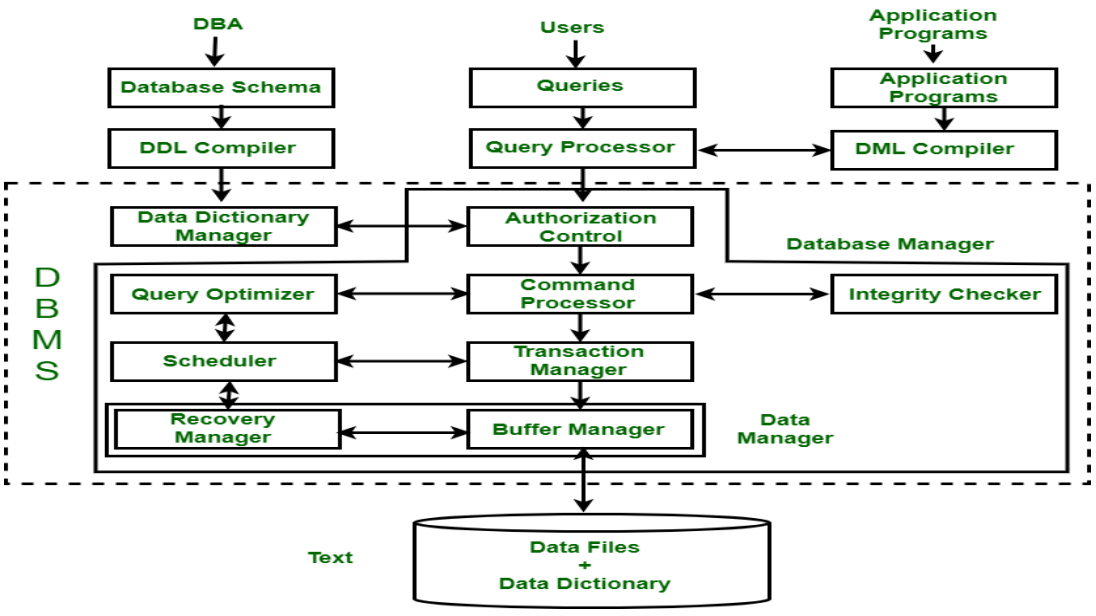
Transaction Control Language-

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit.

Overall Database structure–



DBMS acts as an interface between user and the database. DBMS are very large and typically divided into modules :-

DDL Compiler - Converts DDL statements to a set of tables containing metadata stored in a data dictionary.Metadata information can be the name of files,data items,storage details of each file,mapping information and constraints,etc.

DML Compiler and Query Optimizer - DML compiler translates the Data Manipulation Languages into query Engine instructions. It might also do optimization for query.
Query processor/optimizer translates statements in a query language into low level instructions the database manager understands. (It is used to find an equivalent but more efficient form).

Data Manager - The data manager is the central software component of the DBMS. It is sometimes referred to as the database control system. The data manager is responsible for interfacing with the file system as show. In addition, the tasks of enforcing constraints to maintain the consistency and integrity of the data, as well as its security, are also performed by the data manager.

Data Dictionary - Data Dictionary is a repository of description of data in the database. A data dictionary contains a list of all files in the database, the number of records in each file and the names and types of each field. Most database management systems keep the data dictionary hidden from users to prevent them from accidentally destroying its content.

Functions of the Data Dictionary
1. Defines the data element.
2. Helps in the scheduling.
3. Helps in the control.
4. Permits the various users who know which data is available and how can it be obtained.
5. Helps in the identification of the organizational data irregularity. 6. Acts as a very essential data management tool.
7. Provides with a good standardization mechanism.
8. Acts as the corporate glossary of the ever growing information resource. 9. Provides the report facility, the control facility along with the excerpt facility.

Data Files - It stores the database.

Compiled DML - The DML complier converts the high level Queries into low level file access commands known as compiled DML.

End Users - End Users are the people who interact with the database through applications or utilities. The various categories of end users are:

1. Casual End Users - These Users occasionally access the database but may need different information each time. They use sophisticated database Query language to specify their requests. For example: High level Managers who access the data weekly or biweekly.

2. Native End Users - These users frequently query and update the database using standard types of Queries. The operations that can be performed by this class of users are very limited and effect precise portion of the database.For example: - Reservation clerks for airlines/hotels check availability for given request and make reservations. Also, persons using Automated Teller Machines (ATM's) fall under this category as he has access to limited portion of the database.

3.Standalone end Users/On-line End Users - Those end Users who interact with the database directly via on-line terminal or indirectly through Menu or graphics based Interfaces. Example:-Library Management System.

Unit – 2

Data modelling using the Entity Relationship model ER Model concepts–

- The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

Entity

- An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

Entity Set

- An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Strong Entity Set Weak Entity Set

Strong Entity Set	Weak Entity Set
it has its own primary key.	It does not save sufficient attributes to form a primary Key on its own.
It is represented by a rectangle.	It is represented by a double rectangle.
It contains a primary key represented by an underline.	It contains a Partial Key or discriminator represented by a dashed underline.
The member of strong entity set is called as dominant entity set.	The member of weak entity set is called as subordinate entity set.
The Primary Key is one of its attributes which uniquely Identifies its member.	The Primary Key of weak entity set is a combination of partial Key and Primary Key of the strong entity set.
The relationship between two strong entity set is represent by a diamond symbol.	The relationship between one strong and a weak entity set is represented by a double diamond sign. It is known as identifying relationship.
The line connecting strong entity set with the relationship is single	The line connecting weak entity set with the identifying relationship is double.
Total participation in the relationship may or may not exist.	Total participation in the identifying relationship always exists.

Attributes

- Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

Types of Attributes

- **Simple attribute −** Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
- **Composite attribute −** Composite aΣributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
- **Derived attribute −** Derived aΣributes are the aΣributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.
- **Single-value attribute −** Single-value attributes contain single value. For example − Social_Security_Number.
- **Multi-value attribute −** Mulϴ-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

Relationship

The association among entities is called a relationship. For example, an employee works_at a department, a student .
and Enrolls are called relationships.

Relationship Set

A set of relationships of similar type is called a relationship set. Like relationship too can have attributes. These attributes are called attributes.

Degree of Relationship

The number of participating entities in a relationship defines the degree of the relationship.

- Binary = degree 2
- Ternary = degree 3
- n-ary = degree

Notation for ER Diagram–

Let us now learn how the ER Model is represented by means of an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an

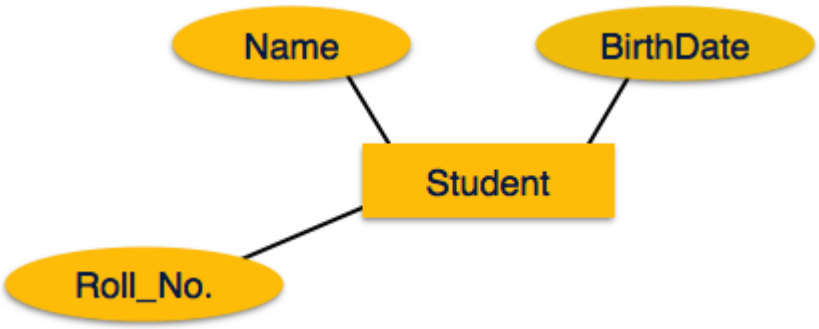
ER diagram.

Entity

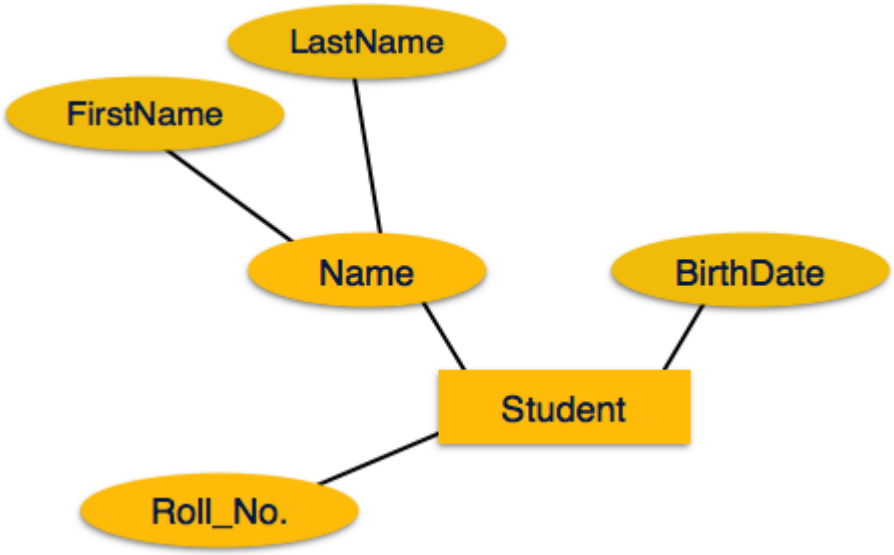
Entities are represented by means of rectangles. Rectangles are named with

Attributes

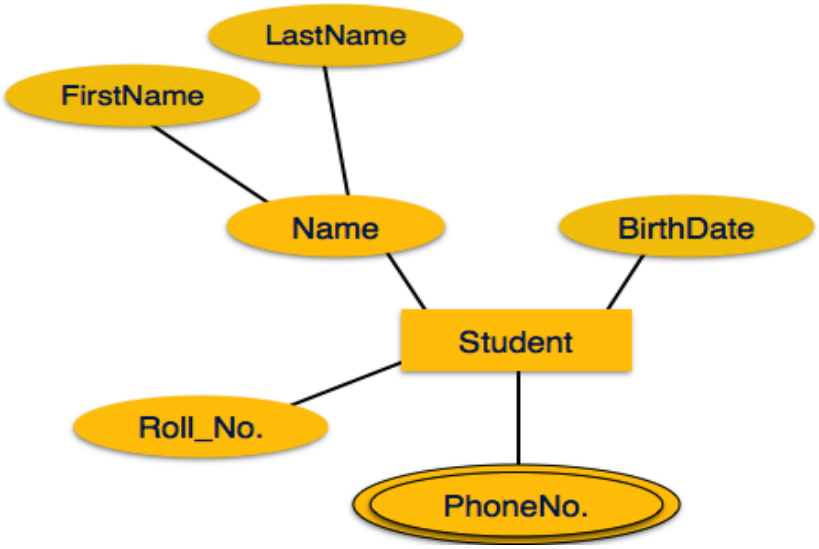
Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



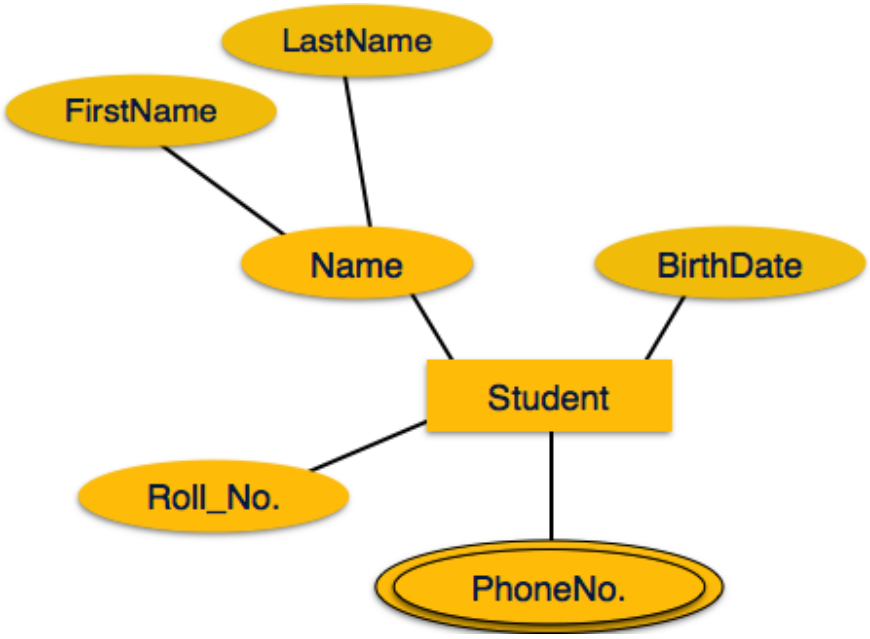
If the attributes are composite, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



Multivalued attributes are depicted by double ellipse.



Derived attributes are depicted by dashed ellipse.



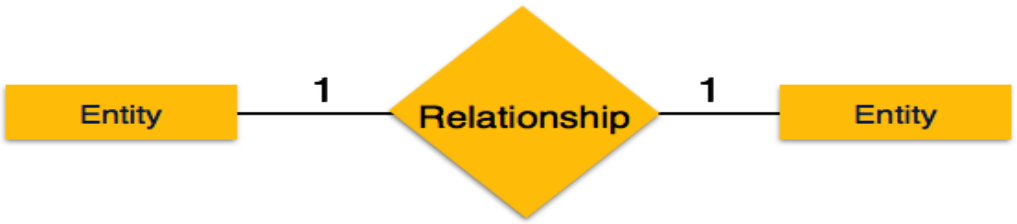
Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

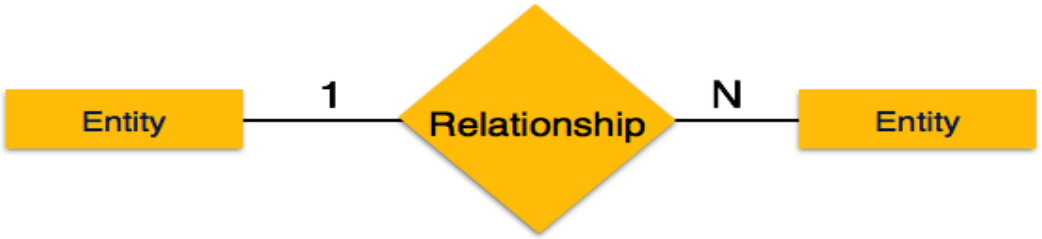
Binary Relationship and Cardinality

A relationship where two entities are participating is called a binary relationship Cardinality is the number of instance of an entity from a relation that associated with the relation.

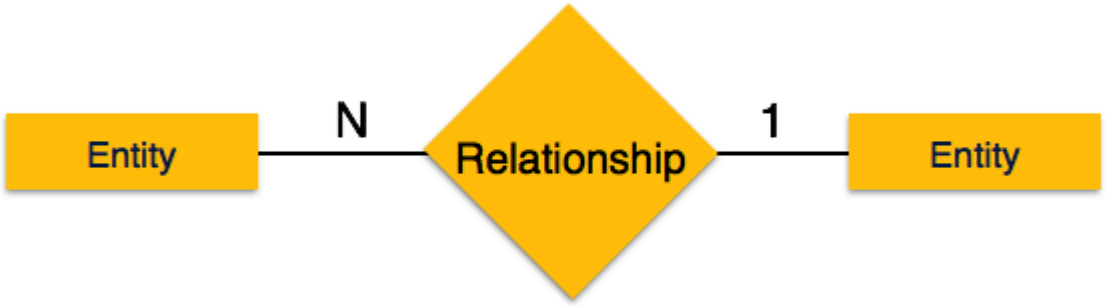
- **One-to-one −** When only one instance of an enΘty is associated with the relationship, it is marked as '1:1'.



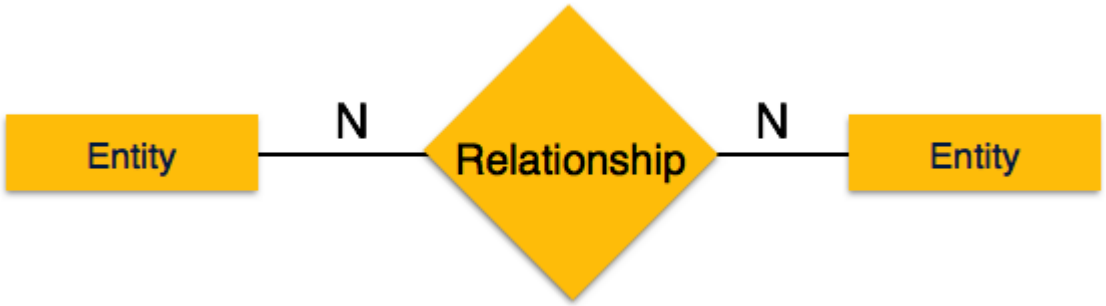
- **One-to-many −** When more than one instance of an enΘty is associated with a relationship, it is marked as '1:N'



- **Many-to-one − −** When more than one instance of enΘty is associated with the relationship, it is marked as 'N:1'.



- **Many-to-many − −** The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.

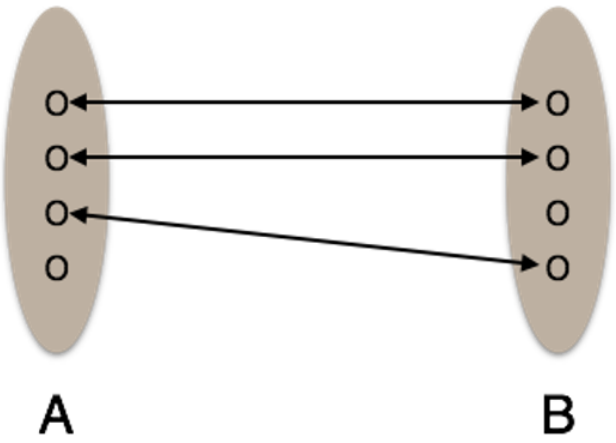


Mapping constraints–

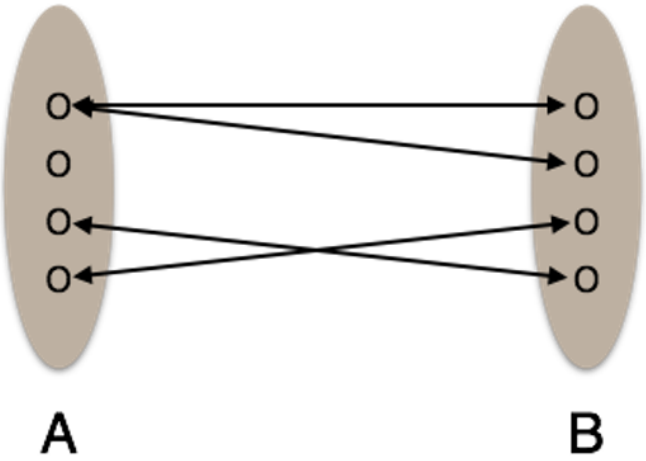
Cardinality or constraints

or constraints defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

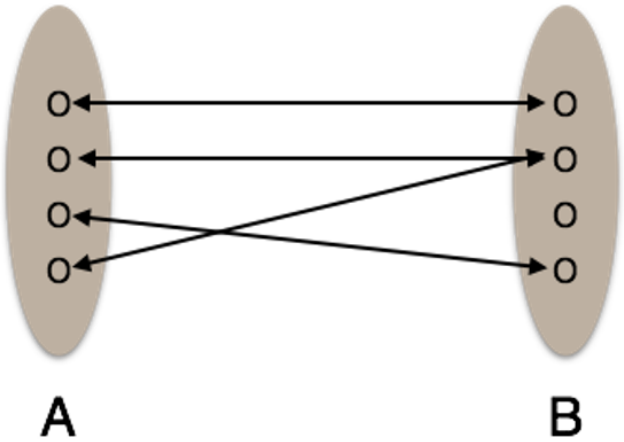
- **One-to-one −** One entity from entity set A can be associated with at most one − One entity from entity set A can be associated with at most one − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



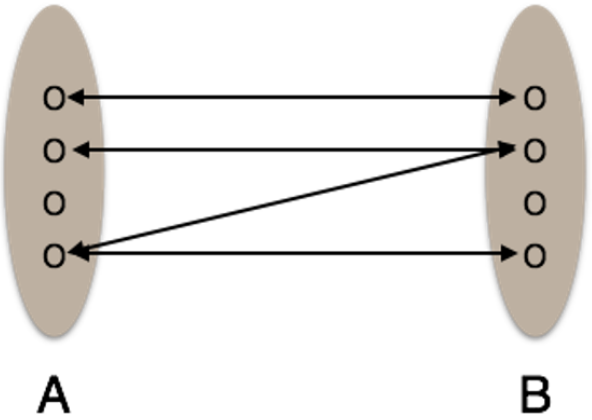
- **One-to-many −** One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be however an entity from entity set B, can be associated with at most one entity.



- **Many-to-one** − More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



- **Many-to-many** − One entity from A can be associated with more than one entity from B and vice versa.



Keys–

- Key plays an important role in relational database; it is used for identifying unique rows from table. It also establishes relationship among tables.

Types of keys in DBMS-

Primary Key – A primary is a column or set of columns in a table that uniquely identifies tuples (rows) in that table.

Super Key – A super key is a set of one of more columns (attributes) to uniquely identify rows in a table.

Candidate Key – A super key with no redundant attribute is known as candidate key

Alternate Key – Out of all candidate keys, only one gets selected as primary key remaining keys are known as alternate or secondary keys.

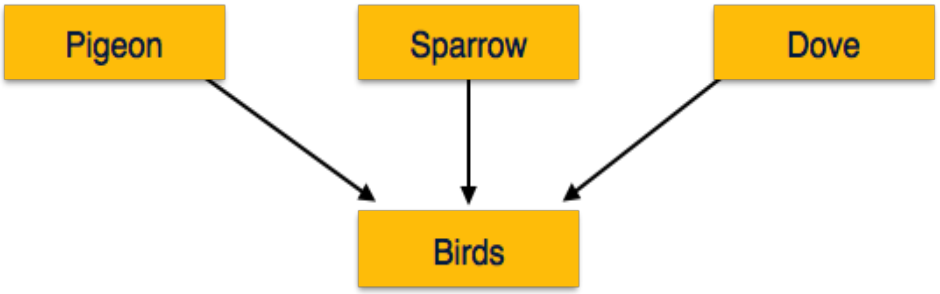
Composite Key – A key that consists of more than one attribute to identify rows (also known as records & tuples) in a table is called composite key.

Foreign Key – Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

Cross

Generalization–

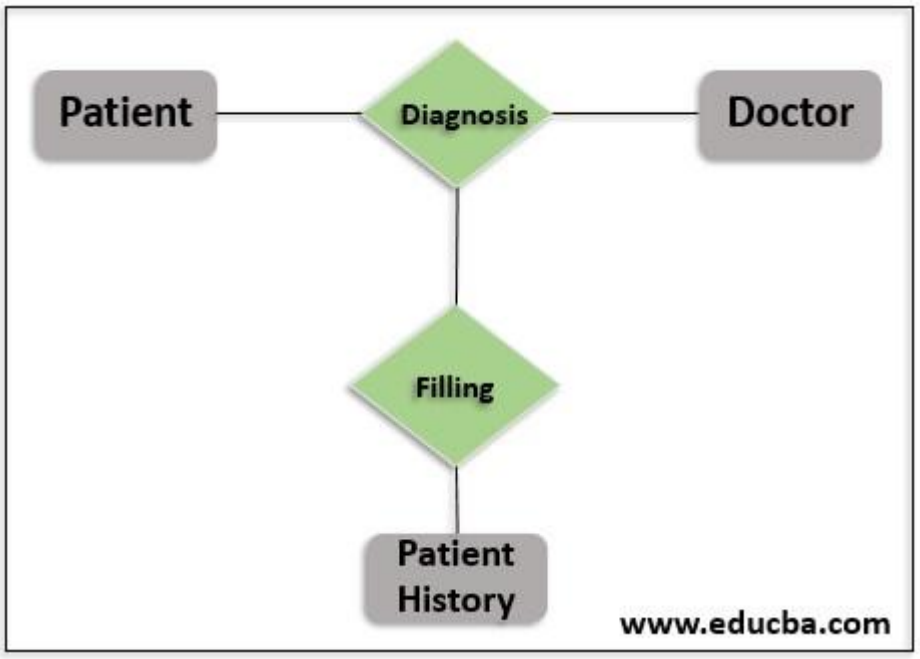
- In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



Aggregation–

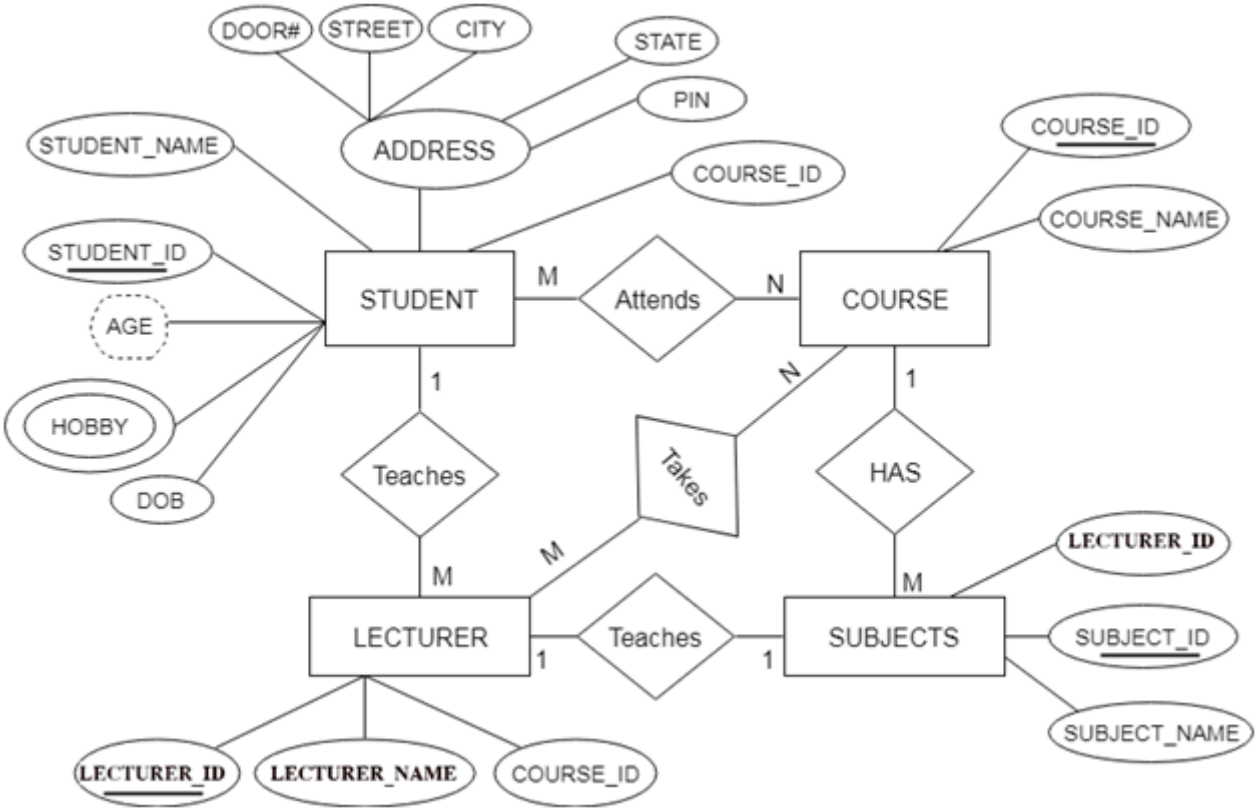
- In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

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 an ER Diagram to table–

The database can be represented using the notations, and these notations can be



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.

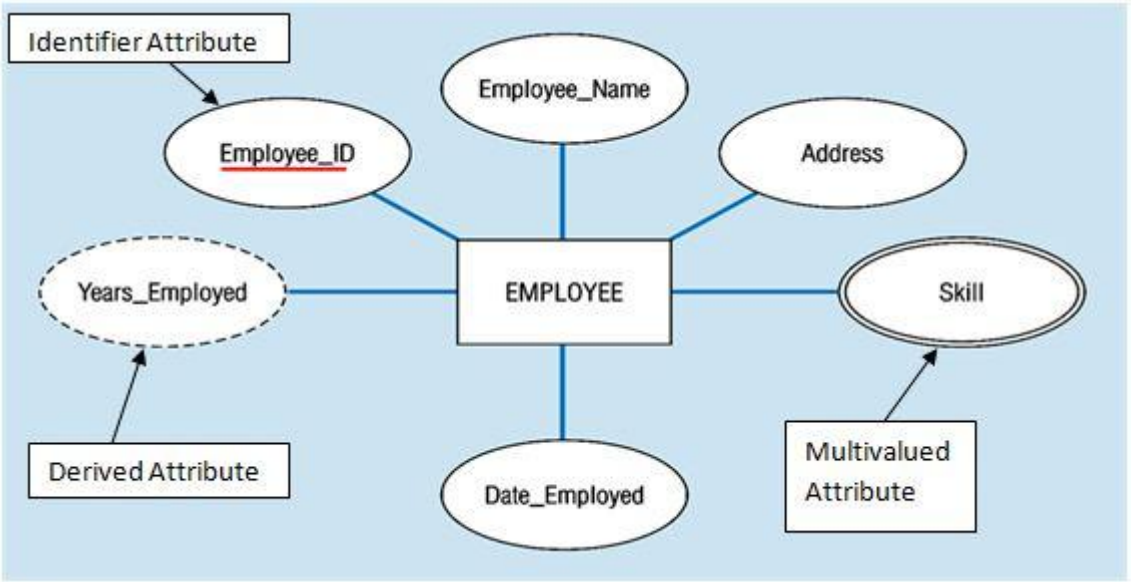
- **Composite attribute represented by components.**

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:



Extended ER Model–

- EER is a high-level data model that incorporates the extensions to the original ER model. It contains the following:
- Subclasses and Super classes
- Specialization and Generalization
- Category or Union type
- Aggregation

Sub class and Super class

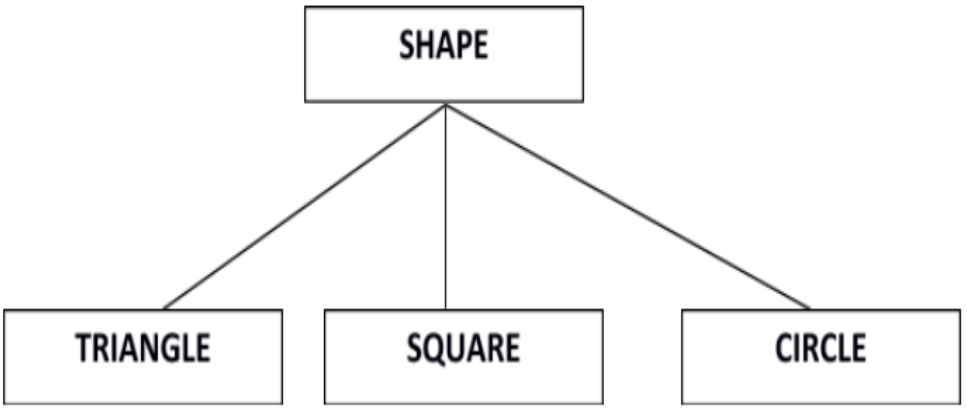
- Sub class and Super class relationship leads the concept of Inheritance.
- The relationship between sub class and super class is denoted with D symbol.

1. Super Class

- Super class is an entity type that has a relationship with one or more subtypes.
- An entity cannot exist in database merely by being member of any super class. For example: Shape super class is having sub groups as Square, Circle, Triangle.

2. Sub Class

- Sub class is a group of entities with unique attributes.
- Sub class inherits properties and attributes from its super class. For example: Square, Circle, Triangle are the sub class of Shape super class.

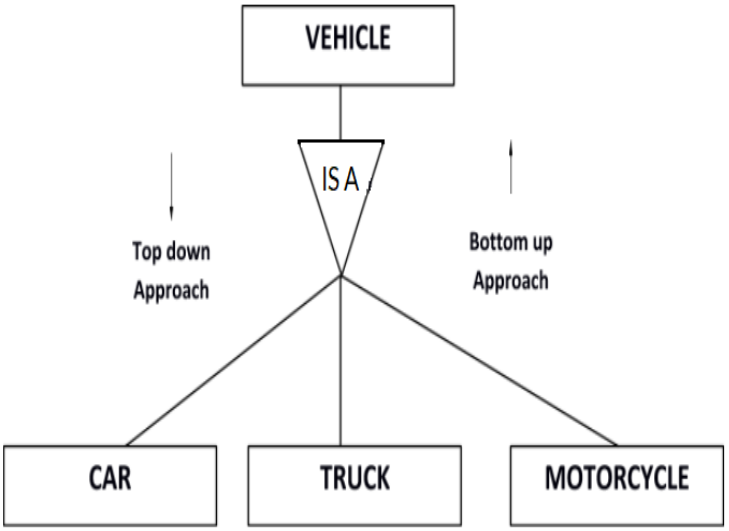


Specialization and Generalization

1. Generalization

- Generalization is the process of generalizing the entities which contain the properties of all the generalized entities.
- It is a bottom approach, in which two lower level entities combine to form a higher level entity.
- Generalization is the reverse process of Specialization.
- It defines a general entity type from a set of specialized entity type.
- It minimizes the difference between the entities by identifying the common features.

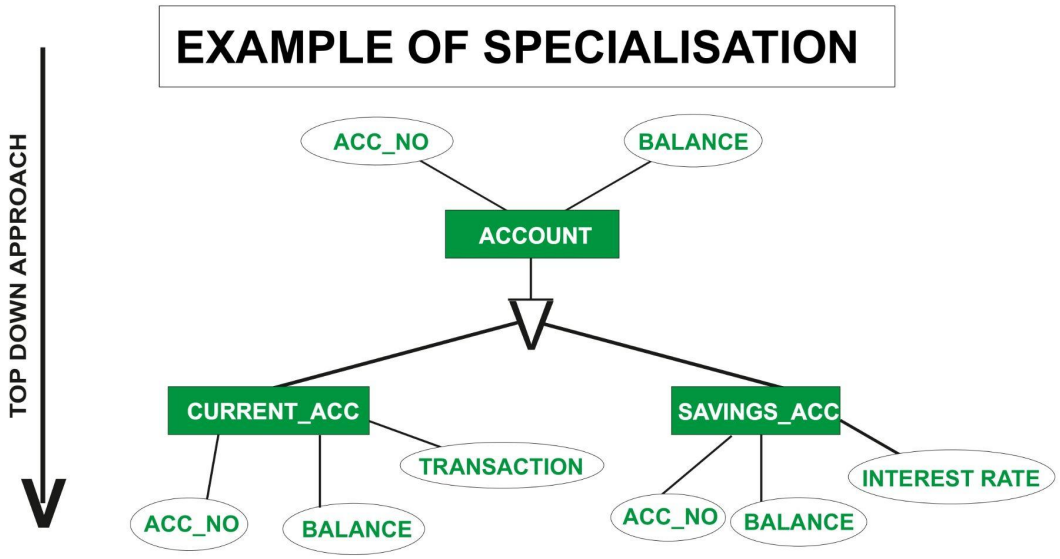
For example:



In the above example, Tiger, Lion, Elephant can all be generalized as

2. Specialization

- Specialization is a process that defines a group entities which is divided into sub groups based on their characteristic.
- It is a top down approach, in which one higher entity can be broken down into two lower level entity.
- It maximizes the difference between the members of an entity by identifying the unique characteristic or attributes of each member.
- It defines one or more sub class for the super class and also forms the superclass/subclass relationship.



Category or Union

- Category represents a single super class or sub class relationship with more than one super class.
- It can be a total or partial participation.

- **For example** Car booking, Car owner can be a person, a bank (holds a possession on a Car) or a company. Category (sub class) → Owner is a subset of the union of the three super classes → Company, Bank, and Person. A Category member must exist in at least one of its super classes.

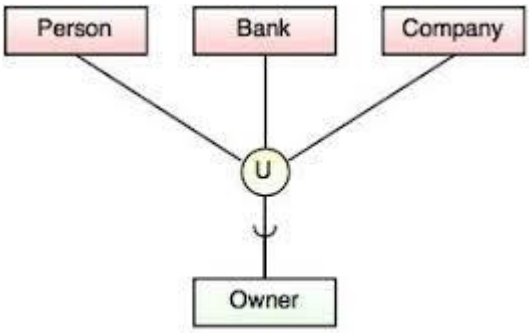
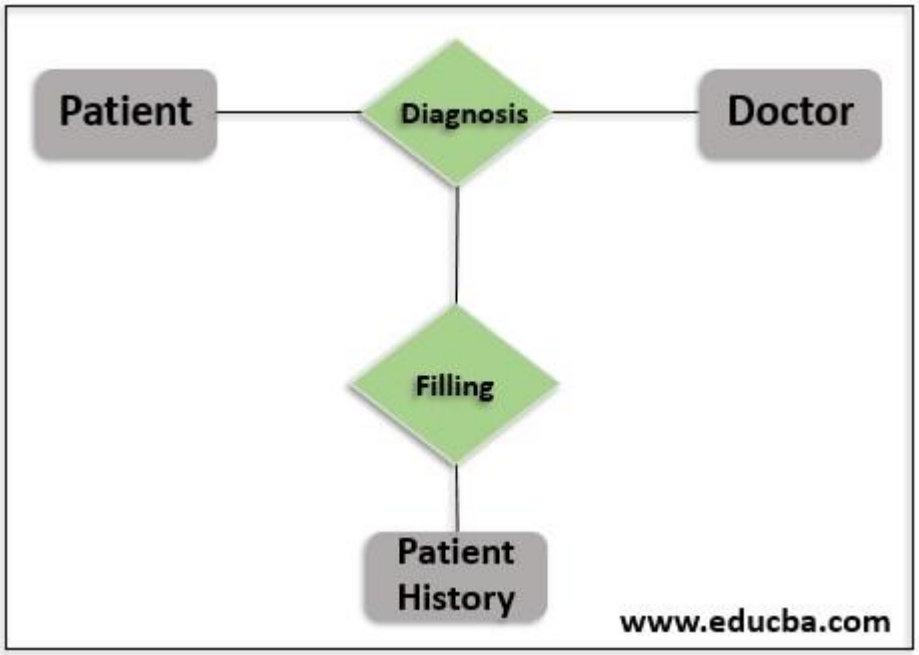


Fig. Categories (Union Type)

Aggregation

- Aggregation is a process that represent a relationship between a whole object and its component parts.
- It abstracts a relationship between objects and viewing the relationship as an object.
- It is a process when two entity is treated as a single entity

.



In the above example, the relation between College and Course is acting as an Entity in Relation with Student.

Unit – 3

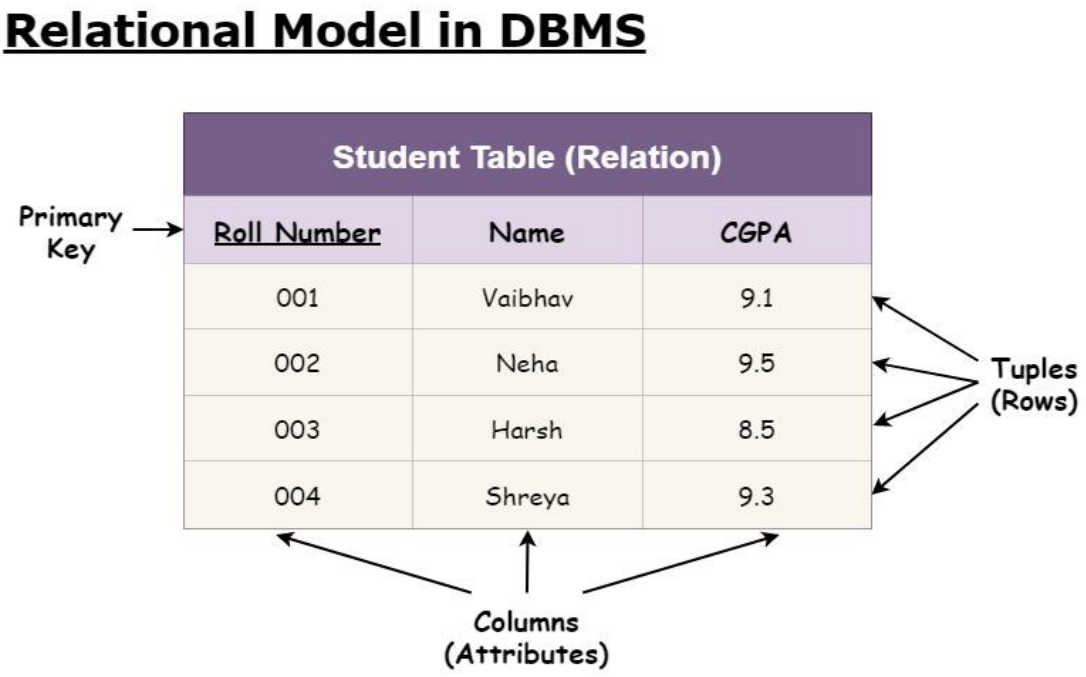
Relational data Model and Language

Relational data model concepts–

- Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

Concepts

- **Tables − In relaϴonal data model, relaϴons are saved in the format of**
- **Tuple −** A single row of a table, which contains a single record for that relation is called a tuple.
- **Relation instance −** A finite set of tuples in the relational database system represents relation instance.
- **Relation schema −** A relaϴon schema describes the relaϴon name (table name), attributes, and their names.
- **Relation key −**which can identify the row in the relation (table) uniquely. which can identify the row in the relation (table) uniquely.



Integrity constraints :-

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called Relational Integrity Constraints. There are three main integrity constraints −

- Key constraints
- Domain constraints
- Referential integrity constraints

Entity integrity–

Referential integrity or Entity integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

Keys constraints–

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called keyfor that relation. If there are more than one such minimal subsets, these are called candidate keys.

Key constraints force that −

- in a relation with a key attribute, no two tuples can have identical values for key attributes.
- a key attribute can not have NULL values.

Example:

- In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

Customer_ID Customer_Name Status 1 Google Active 2 Amazon Active

Domain constraints–

- Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a

specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

Relational algebra–

- Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either unary or binary. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows −

- Select
- Project
- Union
- Set different
- Cartesian product
- Rename

Select Operation (σ)

It selects tuples that satisfy the given predicate from a

relation. **Notation − $\sigma_p(r)$**

Where σ stands for selection predicate and r stands for relation. p is prepositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like − =, ≠, ≥, < , >, ≤.

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation − $\prod_{A1, A2, An}(r)$

Where A_1, A_2 , A_n are attribute names of relation r.

Duplicate rows are automatically eliminated, as relation is

a set. For example −

$\prod_{subject, author}$ (Books)

Selects and projects columns named as subject and author from the relation Books.

Union Operation ( )

It performs binary union between two given relations and is defined as –

$r \cup s = \{ t \mid t \in r \text{ or } t \in s\}$

Notation − r U s

Where r and s are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

- r, and s must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$\prod_{author}(Books) \cup \prod_{author}(Articles)$

Output − Projects the names of the authors who have either written a book or an article or both.

Set Difference (−)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation − r − s

Finds all the tuples that are present in r but not in s.

$\prod_{author}(Books) - \prod_{author}(Articles)$
authors who have wriΣen books but not arΘcles.

Cartesian Product (X)

Combines information of two different relations into one.

Notation − r X s

Where r and s are relations and their output will be defined

as − r X s = { q t | q ∈ r and t ∈ s}

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter rho ρ.

Notation − $\rho_x(E)$

Where the result of expression E is saved with name of x.

Additional operations are −

- Set intersection
- Assignment
- Natural join

Unit – 4

Introduction to SQL

- SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus. SQL comes as a package with all major distributions of RDBMS.

Characteristics of SQL–

- SQL is easy to learn.
- SQL is used to access data from relational database management systems. o SQL can execute queries against the database.
- SQL is used to describe the data.
- SQL is used to define the data in the database and manipulate it when needed.
- SQL is used to create and drop the database and table.
- SQL is used to create a view, stored procedure, function in a database.
- SQL allows users to set permissions on tables, procedures, and views.

Advantages of SQL–

There are the following advantages of SQL:

High speed

- Using the SQL queries, the user can quickly and efficiently retrieve a large amount of records from a database.

No coding needed

- In the standard SQL, it is very easy to manage the database system. It doesn't require a substantial amount of code to manage the database system.

Well defined standards

- Long established are used by the SQL databases that are being used by ISO and ANSI.

Portability

- SQL can be used in laptop, PCs, server and even some mobile phones.

Interactive language

- SQL is a domain language used to communicate with the database. It is also used to receive answers to the complex questions in seconds.

Multiple data view

- Using the SQL language, the users can make different views of the database structure.

SQL Data types and literals–

Data Types

- SQL Data Type is an attribute that specifies the type of data of any object. Each column, variable and expression has a related data type in SQL. SQL Server offers six categories of data types for your use which are listed below –

Exact Numeric Data Types

DATA TYPE	MIN	MAX	STORAGE
Bigint	-9.22337E+18	2^63-1	8 bytes
Int	-2147483648	2147483647	4 bytes
Smallint	-32768	32767	2 bytes
Tinyint	0	255	1 bytes
Bit	0	1	1 to 8 bit columns in the same table requires a total of 1 byte, 9 to 16 bits = 2 bytes, etc...
Decimal	1E+38	10^38–1	Precision 1-9 = 5 bytes, precision 10-19 = 9 bytes, precision 20-28 = 13 bytes, precision 29-38 = 17 bytes
Numeric	same as Decimal	same as Decimal	same as Decimal
Money	-9.22337E+14	2^63-1 / 10000	8 bytes
Smallmoney	-214748.3648	214748.3647	4 bytes

Approximate Numeric Data Types

TABLE 3-3 Approximate Numeric Data Types

DATA TYPE	STORAGE SIZE	POSSIBLE VALUES
float (n <= 24)	4 bytes	–3.40E38 to –1.18E-38, 0 and 1.18E-38 to 3.40E38
float (24 > n <= 53)	8 bytes	–1.79E308 to –2.23E-308, 0 and 2.23E-308 to 1.79E308
real	Functionally equivalent to *float(24)*	

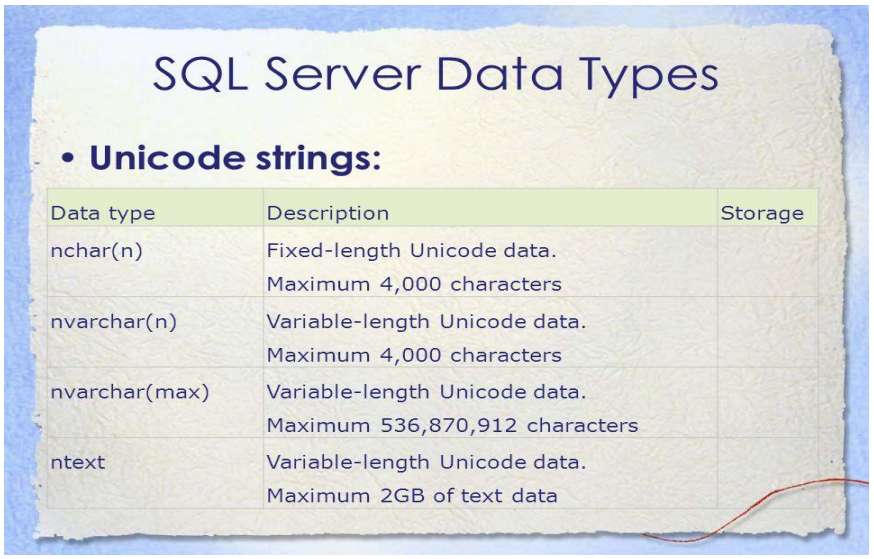
Date and Time Data Types

Data Type	Storage (bytes)	Date Range	Accuracy
DATETIME	8	January 1, 1753 to December 31, 9999	3-1/3 milliseconds
SMALLDATETIME	4	January 1, 1900 to June 6, 2079	1 minute
DATETIME2	6 to 8	January 1, 0001 to December 31, 9999	100 nanoseconds
DATE	3	January 1, 0001 to December 31, 9999	1 day
TIME	3 to 5		100 nanoseconds
DATETIMEOFFSET	8 to 10	January 1, 0001 to December 31, 9999	100 nanoseconds

Character Strings Data Types

Data Type	Notes
char	Fixed length single-byte character data
nchar	Fixed length Unicode character data
varchar	Variable length single-byte character data
nvarchar	Variable length Unicode character data
varchar(max)	Large value single-byte character data
nvarchar(max)	Large value Unicode character data
text	Variable length single-byte character data - do not use
ntext	Variable length Unicode data - do not use

Unicode Character Strings Data Types



Data Literals

- An program source element that represents a data value. Data literals can be divided into multiple groups depending the type of the data it is representing and how it is representing.

1. Character String Literals are used to construct character strings, exact numbers, approximate numbers and data and time values. The syntax rules of character string literals are pretty simple:

- A character string literal is a sequence of characters enclosed by quote characters.
- The quote character is the single quote character "'".
- If "'" is part of the sequence, it needs to be doubled it

  as "''". Examples of character string literals:

Quote:

```
'Hello world!'
'Loews L''Enfant Plaza'
'123'
'0.123e-1'
'1999-01-01'
```

2. Hex String Literals are used to construct character strings and exact numbers. The syntax rules for hex string literals are also very simple:

- A hex string literal is a sequence of hex digits enclosed by quote characters and prefixed with "x".
- The quote character is the single quote character "'".

Examples of hex string literals: **Quote**:

```
X'015'
X'01aF'
x'BC'
x'2d'
0x7e4
0x88bA
```

3. Numeric Literals are used to construct exact numbers and approximate numbers. Syntax rules of numeric literals are:

- A numeric literal can be written in signed integer form, signed real numbers without exponents, or real numbers with exponents.

Examples of numeric literals:

Quote:

```
1
-22
33.3
-44.44
55.555e5
-666.666e-6
```

4. Date and Time Literals are used to construct date and time values. The syntax of date and time literals are:

- A date literal is written in the form of "DATE 'yyyy-mm-dd'".
  A time literal is written in the form of "TIMESTAMP 'yyyy-mm-dd hh:mm:ss'".

Examples of data and time literals:

Quote:

```
DATE '1999-01-01'
TIMESTAMP '1999-01-01 01:02:03'
```

Types of SQL Commands–

SQL commands are mainly categorized into four categories as discussed below:

1. **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 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.

2. **DML(Data Manipulation Language) :** The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements. **Examples of DML:**
   - **SELECT –** is used to retrieve data from the a database.
   - **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.

3. **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.

4. **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

SQL Operators and their procedure–

SQL statements generally contain some reserved words or characters that are used to perform operations such as comparison and arithmetical operations etc. These reserved words or characters are known as operators.

Generally there are three types of operators in SQL:

1. SQL Arithmetic Operators
2. SQL Comparison Operators
3. SQL Logical Operators

SQL Arithmetic Operators:

Let's assume two variables "a" and "b". Here "a" is valued 50 and "b"

Operators:

SQL supports various arithmetic, logical , character, and relational r operators.

Arithmetic Operators:

Arithmetic operators in SQL will return numeric values.

Operators	Function	Example
+	Add	Update set emp basic = basic + 500;
-	Subtract	Update set emp basic = basic - 500;
*	Multiply	Update set emp basic = basic + basic * 5/100;
/	Divide	Update set emp basic = basic + basic * 5/100;

SQL Comparison Operators:

Index	Operator	Description
1	=	Equality operator or Equal to
2	!=	Inequality operator or Not equal to
3	<>	Inequality operator or Not equal to
4	<=	Less than or Equal to
5	>=	Greater than or Equal to
6	<	Less than operator
7	>	Greater than operator
8	!>	Not greater than operator
9	!<	Not less than operator

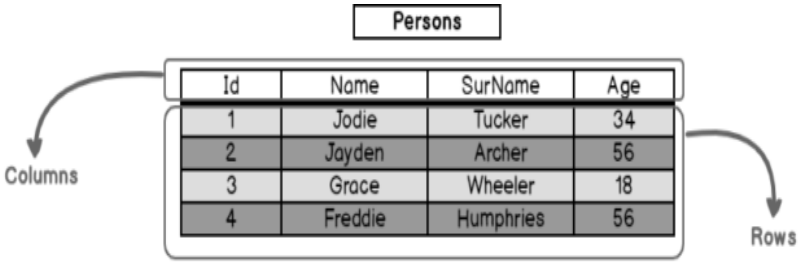
SQL Logical Operators:

This is the list of logical operators used in SQL.

Operator	Meaning
ALL	TRUE if all of a set of comparisons are TRUE.
AND	TRUE if both Boolean expressions are TRUE.
ANY	TRUE if any one of a set of comparisons are TRUE.
BETWEEN	TRUE if the operand is within a range.
EXISTS	TRUE if a subquery contains any rows.
IN	TRUE if the operand is equal to one of a list of expressions.
LIKE	TRUE if the operand matches a pattern.
NOT	Reverses the value of any other Boolean operator.
OR	TRUE if either Boolean expression is TRUE.
SOME	TRUE if some of a set of comparisons are TRUE.

Tables–

- Tables are made up of rows and columns and the intersection of rows and columns made cell.
- The name of the column is known as the attributes and the row is known as tupples.



Queries and subqueries–

- A query is a question, often expressed in a formal way. A database query can be either a select query or an action query. A select query is a data retrieval query, while an action query asks for additional operations on the data, such as insertion, updating or deletion.

Subquery

- A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause. ▪ A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved. ▪ Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

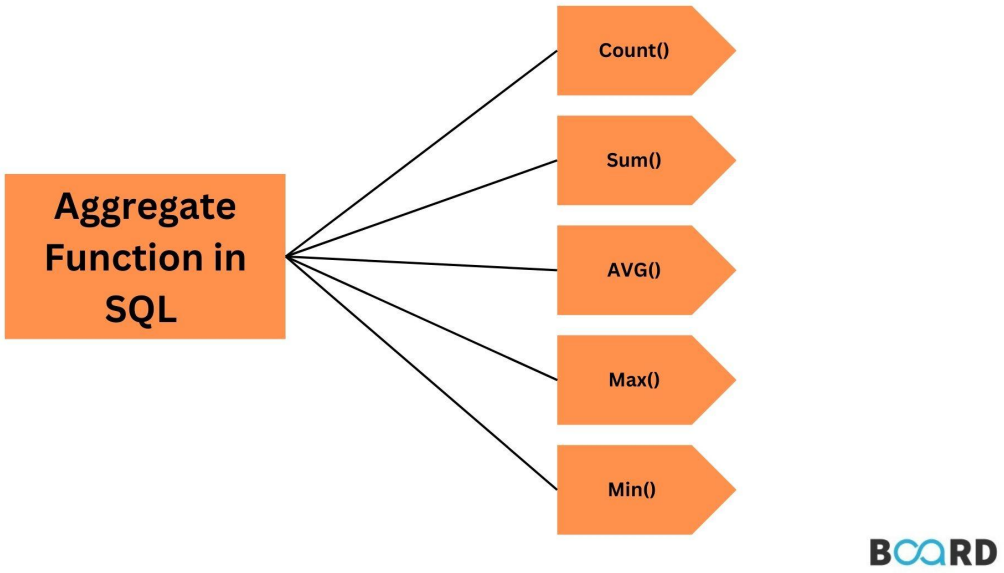
Aggregate functions–

- The aggregate function allow us to perform a calculations on a set of values. The following are the most commonly SQL aggregate

functions

1. AVG – calculates the average of a set of values.

Example:



Insert–

- Insert command is used to insert data into a created table. **Example:**

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Update and Delete operations–

- SQL UPDATE Statement is used to update data from the table. **Example:**

```
DELETE FROM table_name WHERE condition;
```

Joins–

- The SQL join clause is use to combine two or more tables in a database.
- There are various types of join available in SQL
  1. Theta join
  2. Equi join
  3. Natural join
  4. Outer join

Theta (θ) Join -

Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol θ.

Notation

$R1 \bowtie_\theta R2$

R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. ,Bn) such that the attributes don't have anything in common, that is $R1 \cap R2 = \Phi$.

Theta join can use all kinds of comparison operators.

Student

S_id	Name	Std	Age
1	Andrew	5	25
2	Angel	10	30
3	Anamika	8	35

Course

Class	C_name
10	Foundation C
5	C++

Student $\bowtie_\theta$ Course

S_id	Name	Std	Age	Class	C_name
1	Andrew	5	25	5	C++
2	Angel	10	30	10	Foundation C

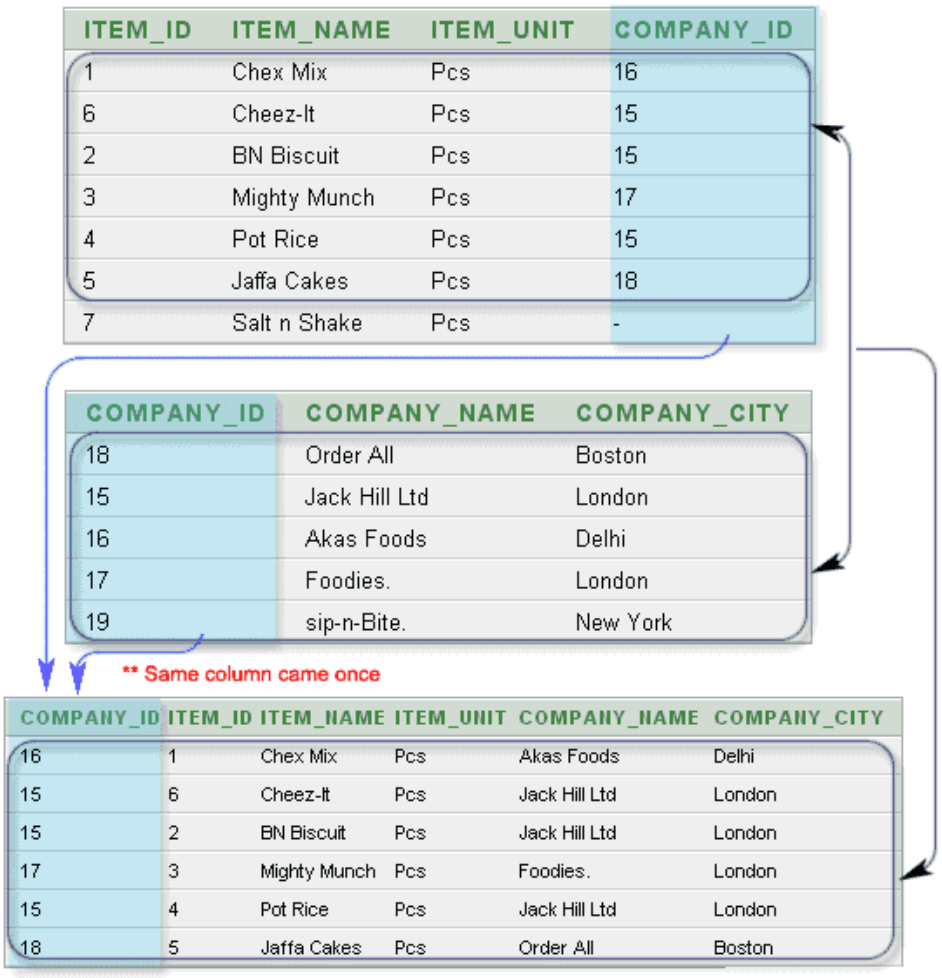
Equijoin -

When Theta join uses only **equality** comparison operator, it is said to be equijoin. The above example corresponds to equijoin.

Natural Join -

Natural join does not use any comparison operator. It does not concatenate the way a Cartesian product does. We can perform a Natural Join only if there is at least one common attribute that exists between two relations. In addition, the attributes must have the same name and domain.

Natural join acts on those matching attributes where the values of attributes in both the relations are same.



ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18
7	Salt n Shake	Pcs	-

COMPANY_ID	COMPANY_NAME	COMPANY_CITY
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

** Same column came once

COMPANY_ID	ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_NAME	COMPANY_CITY
16	1	Chex Mix	Pcs	Akas Foods	Delhi
15	6	Cheez-It	Pcs	Jack Hill Ltd	London
15	2	BN Biscuit	Pcs	Jack Hill Ltd	London
17	3	Mighty Munch	Pcs	Foodies.	London
15	4	Pot Rice	Pcs	Jack Hill Ltd	London
18	5	Jaffa Cakes	Pcs	Order All	Boston

Outer join –

There are three kinds of outer joins leÕ outer join, right outer join, and full

Left
OuterJoin(R S)

All the tuples from the Left relation, R, are included in the resulting relation. If there are tuples in R without any matching tuple in the Right relation S, then the

S-attributes of the resulting relation are made NULL.

A.TABLEMASTER

Date	Name	Location	Doctor	drugDescription	Insurance
5-01-2018	antra	mumbai	Rolf	amoxicillin 400gm	abc
2-02-2018	priya	UK	Jake	Lisinopril 10g	kotack
13-03-2018	riya	australia	Sofia	Hydrocodone 40gm	tef
16-04-2018	seema	china	Indira	Abreva 34gm	sdc

B.TABLE1

Date	drug	Price	Code
5-01-2018	Lisinopril	23.67	AB34
2-02-2018	Hydrocodone	45	CD12
13-03-2018	Abreva	489	KD23

C.TABLE2

codefull	DrugDescription	Unit	Calculation type	drug
AB35JIUO	Lisinopril 10g	1	multiply	Lisinopril
CD80S23D	Hydrocodone 40gm	1	multiply	Hydrocodone
KD896F45	Abreva 34gm	1	divide	Abreva

Right Outer Join: (R S)

All the tuples from the Right relation, S, are included in the resulting relation. If there are tuples in S without any matching tuple in R, then the R-attributes of resulting relation are made NULL.

```
Syntax1:
SELECT column_name(s)
FROM table1
RIGHT OUTER JOIN table2
ON table1.column_name = table2.column_name;

Syntax2:
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Full Outer Join: (R S)

All the tuples from both participating relations are included in the resulting relation. If there are no matching tuples for both relations, their respective unmatched attributes are made

```
SELECT O.OrderID, O.OrderDate, O.BookID, B.Title, B.Auth
FROM Orders AS O
LEFT OUTER JOIN Books AS B
ON B.BookID = O.BookID
```

Results | Messages

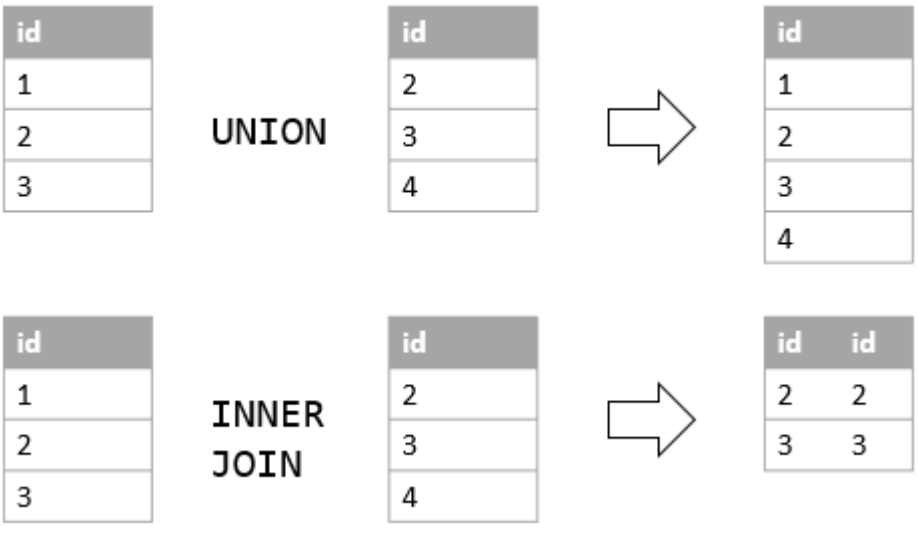
OrderID	OrderDate	BookID	Title	Author
100	2021-04-01	4	American Fire	Hesse
101	2021-04-06	3	T-SQL Fundamentals	Ghan
102	2021-04-23	2	Eat that frog!	Tracy
103	2021-04-30	6	Pro SQL Server Internals	Korotkevitch
104	2021-05-06	4	American Fire	Hesse
105	2021-05-15	5	Atomic Habits	Clear
106	2021-05-17	17	NULL	NULL

Unions–

Union merges the results of two SELECT statements.

For example:

Simple Table



Unit – 5
Database Design and Normalization

Functional dependencies–

- Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.
- Functional dependency is represented by an arrow sign (→) that is, X→Y, where X funcϴonally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

Normal Forms–

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

- **Update anomalies −** If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
- **Deletion anomalies −** We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
- **Insert anomalies −** We tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring the database to a consistent state.

First– First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values

defines that all the attributes in a relation must have atomic domains. The
values in an atomic domain are indivisible units.

ID	Name	DOB
3	FRANK	28-06-1996
2	ADELYN	06-09-1995
4	JULIUS	13-12-1996
1	MADDEN	05-04-1997
5	MARKER	06-06-1997

5 row in set (0.00 sec)

Second–

Before we learn about the second normal form, we need to understand the following −

TABLE_PURCHASE

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

TABLE_STORE

Store ID	Purchase Location
1	Los Angeles
2	New York
3	San Francisco

Third–

For a relation to be in Third Normal Form, it must be in Second Normal form and For a relation to be in Third Normal Form, it must be in Second Normal form and

For a relation to be in Third Nor

Third Normal Form - Example

PART_ID

IDENT	NAME	CITY
P1	Collins	London
P2	Jones	Glasgow
P3	Rodin	Aberdeen
P4	Thatcher	London
P5	Biggs	Bristol

CITIES

CITY	INHAB
London	8000000
Glasgow	400000
Aberdeen	400000
Bristol	800000

PART_COURSE

IDENT	COURSE	GRADE
P1	English	A
P1	Geography	C
P1	Logic	A
P2	Geography	B
P2	Database	C
P3	Physics	B
P4	Logic	A
P4	Chemistry	C
P5	Database	A
P5	English	A
P5	Biology	A