UNIT - I

Introduction:

A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner. It allows users to create, modify, and query a database, as well as manage the security and access controls for that database. DBMS provides an environment to store and retrieve data in convenient and efficient manner.

Characteristics of DBMS

- **Data modeling:** A DBMS provides tools for creating and modifying data models, which define the structure and relationships of the data in a database.
- **Data storage and retrieval:** A DBMS is responsible for storing and retrieving data from the database, and can provide various methods for searching and querying the data.
- **Concurrency control:** A DBMS provides mechanisms for controlling concurrent access to the database, to ensure that multiple users can access the data without conflicting with each other.
- **Data integrity and security:** A DBMS provides tools for enforcing data integrity and security constraints, such as constraints on the values of data and access controls that restrict who can access the data.
- **Backup and recovery:** A DBMS provides mechanisms for backing up and recovering the data in the event of a system failure.

File system vs database approach:

There are the following differences between DBMS and File systems:

DBMS Approach	File System Approach
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.
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.
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.
DBMS provides a good protection mechanism.	It isn't easy to protect a file under the file system.
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.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	The 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 deleting some information or updating some information.

Database approach used in large systems which interrelate many files.	File system approach used in large systems which interrelate many files.
The database system is expensive to design.	The file system approach is cheaper to design.
Due to the centralization of the database, the problems of data redundancy and inconsistency are controlled.	In this, the files and application programs are created by different programmers so that there exists a lot of duplication of data which may lead to inconsistency.
The database structure is complex to design.	The file system approach has a simple structure.
In this system, Data Independence exists, and it can be of two types. ○ Logical Data Independence ○ Physical Data Independence	In the File system approach, there exists no Data Independence.
Integrity Constraints are easy to apply.	Integrity Constraints are difficult to implement in file system.
In the database approach, 3 types of data models exist: ○ Hierarchal data models ○ Network data models ○ Relational data models	In the file system approach, there is no concept of data models exists.
Changes are often a necessity to the content of the data stored in any system, and these changes are more easily with a database approach.	The flexibility of the system is less as compared to the DBMS approach.
Oracle, SQL Server, Sybase etc.	Notpad,word files

Different Types of Database Users

A Database User is defined as a person who interacts with data daily, updating, reading, and modifying the given data. Database users can access and retrieve data from the database through the Database Management System (DBMS) applications and interfaces.

Types of Database Users

Database users are categorized based on their interaction with the database. There are seven types of database users in DBMS. Below mentioned are the types of database users:

1. Database Administrator (DBA)
Database Administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of the database. The DBA will then create a new account ID and password for the user if he/she needs to access the database. DBA is also responsible for providing security to the database and he allows only authorized users to access/modify the database. DBA is responsible for problems such as security breaches and poor system response time.

- DBA also monitors the recovery and backup and provides technical support.
- The DBA has a DBA account in the DBMS which is called a system or superuser account.
- DBA repairs damage caused due to hardware and/or software failures.
- DBA is the one having privileges to perform DCL (Data Control Language) operations such as GRANT and REVOKE, to allow/restrict a particular user from accessing the database.

2. Naive / Parametric End Users
Parametric End Users are the unsophisticated who don't have any DBMS knowledge but they frequently use the database applications in their daily life to get the desired results. For example, Railway's ticket booking users are naive users. Clerks in any bank is a naive user because they don't have any DBMS knowledge but they still use the database and perform their given task.

3. Sophisticated Users
Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database. They can develop their own database applications according to their requirement. They don't write the program code but they interact the database by writing SQL queries directly through the query processor.

4. Application Programmers
Application Programmers also referred as System Analysts or simply Software Engineers, are the front-end programmers who writes the code for the application programs. They are the computer professionals. These programs could be written in Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL ,java etc

Advantages of Database Management System (DBMS)

Some of the great advantages of DBMS are listed below:

1. **Better use of data or information** - We can easily and efficiently access well-managed and synchronized forms of data with the help of DBMS. It makes data handling simple, provides an integrated perspective of how a certain business is operating and also aids in keeping track of how one element of the business affects another portion.
2. **Secured Data** - The likelihood of security problems increases as a database becomes more functional and accessible. The danger to data security rises as a result of the rate at which data is shared or transferred growing along with the user base. It is frequently utilized in the business world where organizations spend a lot of time, money, and effort making sure data is protected and handled effectively. Data management systems (DBMS) offer a stronger framework for data privacy and security policies, assisting businesses in enhancing data security.
3. **Reduces Data Inconsistency and Redundancy** - The major issues faced during the process of storing data are inconsistency and redundancy. Inconsistent data may lead to a big loss to an individual or a business model and the storage capacity is not utilized

properly because of the data redundancy. When multiple copies with different versions or values of the same data exist in various locations, then it causes inconsistency. Data Redundancy and inconsistency can both be significantly decreased by properly designing a database with the help of a database management system.

4. **Better Recovery and Backups** - Backup and recovery are handled automatically by the DBMS. Users don't need to regularly back up their data because the DBMS handles this for them. Additionally, it returns the database to its prior state following a crash or system failure.
5. **Fast Data Sharing** - Database administration makes it possible for consumers to access more and better-managed data. DBMS enables end users to quickly scan their environment and react to any alterations made there.
6. **Helps in decision-making** - Because of the well-managed data and improved data access provided by DBMS, we are able to produce better-quality information and, as a result, make better judgments. Accuracy, validity, and the time it takes to read data are all improved by better data quality. Although DBMS does not ensure data quality, it does offer a framework that makes it simple to enhance data quality.
7. **Increases Privacy** - The privacy rule in a database specifies the privacy restrictions that can only be accessed by authorized users. A user can only view the data he is permitted to view since there are different degrees of database access. For instance, on social networking sites, different accounts that a user wishes to access have varying access restrictions and a user can only see his/her account details, not others.
8. **User Friendly** - Data are presented in a straightforward and logical manner by database management systems (DBMS). It is simple to carry out many activities, such as the addition, deletion, or creation of files or data.
9. **Data Abstraction** - In order to give users an abstract overview of the data, database systems are primarily used. Since numerous intricate algorithms are employed by developers to boost the effectiveness of databases that are concealed from users by several degrees of data abstraction, consumers can easily engage with the system.

Database Applications

Nowadays, any business that has small or large amounts of data needs a database to store and manage the information. The database is an easy, reliable, secure, and efficient way to maintain business information. There are many applications where databases are used.

1. Universities:

It is an undeniable application of the database. Universities have so much data which can be stored in the database, such as student information, teacher information, non-teaching staff information, course information, section information, grade report information, and many more. University information is kept safe and secure in the database.

2. Banking:

It is one of the major applications of databases. Banks have a huge amount of data as millions of people have accounts that need to be maintained properly. The database keeps the record of each user in a systematic manner. Banking databases store a lot of information about account holders. It stores customer details, asset details, banking transactions, balance sheets, credit card and debit card details, loans, fixed deposits, and much more. Everything is maintained with the help of a database.

3. Railway Reservation System:
It is an inevitable area of application of databases. They store information such as passenger name, mobile number, booking status, reservation details, train schedule, employee information, account details, seating arrangement, route & alternate route details, etc. All the information needs to be maintained, so railways use a database management system for their efficient storage and retrieval purpose.

4. Social Media Sites:
Nowadays, everyone has a smartphone and accounts on various social media sites like Facebook, LinkedIn, Pinterest, Twitter, Instagram, etc. People can chat with their friends and family and make new friends from all over the world. Social media has millions of accounts, which means they have a huge amount of data that needs to be stored and maintained. Social media sites use databases to store information about users, images, videos, chats, etc.

5. Library Management System:
There are hundreds and thousands of books in the library, so it is not easy to maintain the records of the books in a register or diary, so a database management system is used which maintains the information of the library efficiently. The library database stores information like book name, issue date, author name, book availability, book issuer name, book return details, etc.

Data Independence

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.

Three Schema Architecture Of Dbms

The three levels present in this architecture are Physical level, Conceptual level and External level.
The details of these levels are as follows −

Physical Level

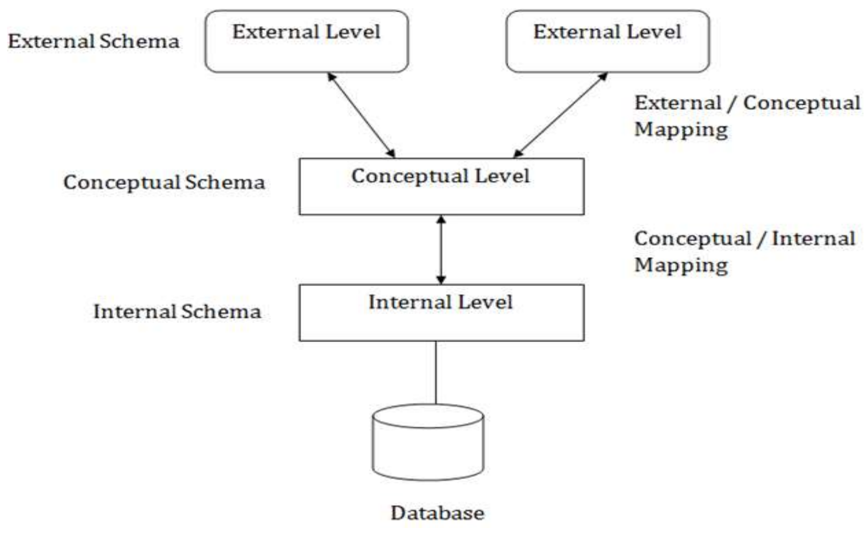
This is the lowest level in the three level architecture. It is also known as the internal level. The physical level describes how data is actually stored in the database. In the lowest level, this data is stored in the external hard drives in the form of bits and at a little high level, it can be said that the data is stored in files and folders. The physical level also discusses compression and encryption techniques.

Conceptual Level

The conceptual level is at a higher level than the physical level. It is also known as the logical level. It describes how the database appears to the users conceptually and the relationships between various data tables. The conceptual level does not care for how the data in the database is actually stored.

External Level

This is the highest level in the three level architecture and closest to the user. It is also known as the view level. The external level only shows the relevant database content to the users in the form of views and hides the rest of the data. So different users can see the database as a different view as per their individual requirements



DATA MODEL

•Data Model gives us an idea that how the final system will look like after its complete implementation. It defines the data elements and the relationships between the data elements.

•Data Models are used to show how data is stored, connected, accessed and updated in the database management system. Here, we use a set of symbols and text to represent the information so that members of the organisation can communicate and understand it.

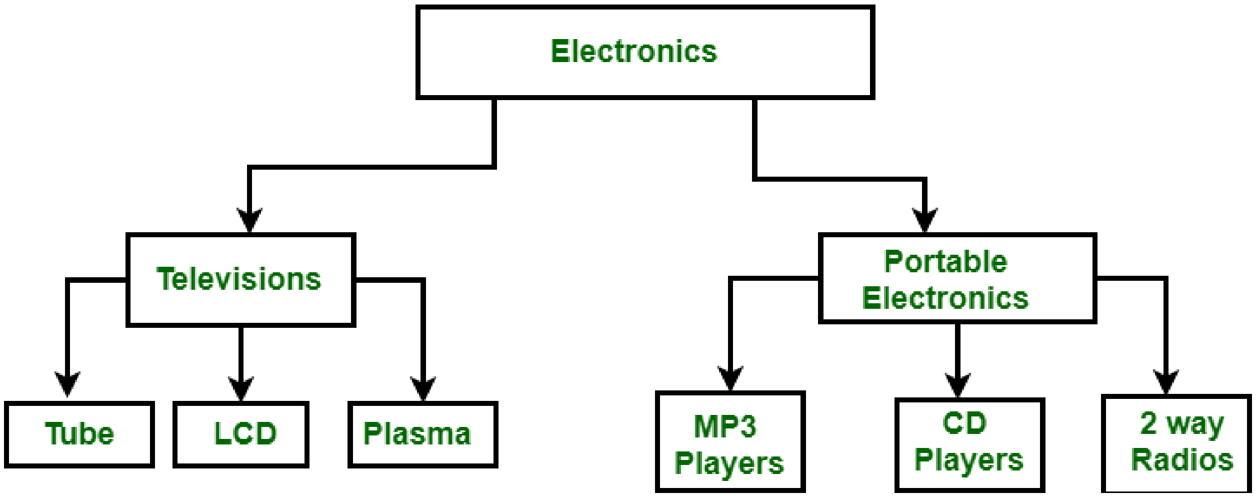
•Though there are many data models being used nowadays but the Relational model is the most widely used model.

1. Hierarchical Model
2. Network Model
3. Entity-Relationship Model
4. Relational Model
5. object oriented data model

Hierarchical Model

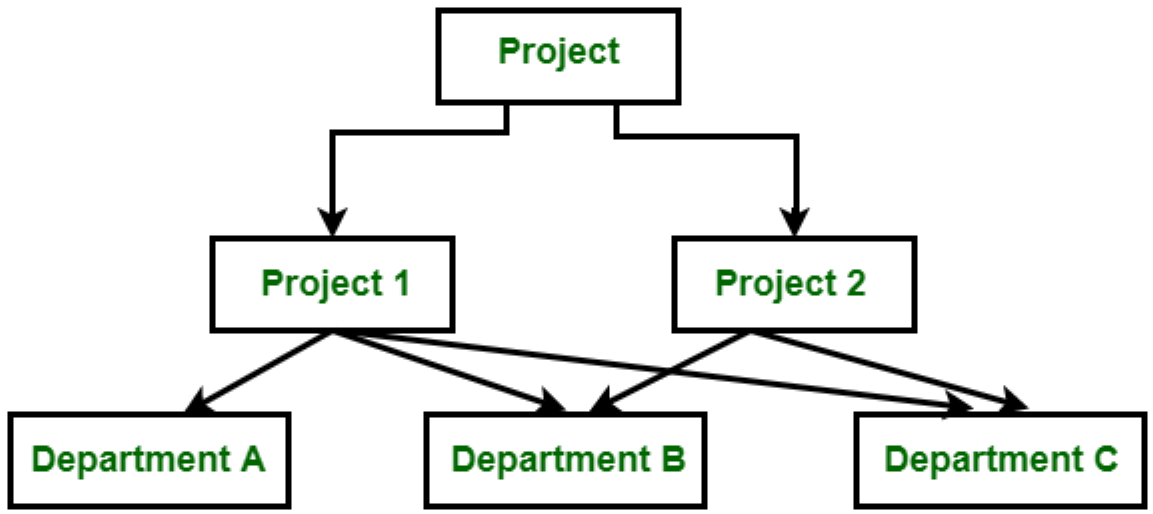
Hierarchical Model was the first DBMS model. This model organises the data in the hierarchical tree structure. The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node.

It is also known as ONE –to-MANY relationship data model.This model defines the one parent data can have more than one child data but the child data can have only one parent data



Network Model

This model is an extension of the hierarchical model. It was the most popular model before the relational model. This model is the same as the hierarchical model, the only difference is that a record can have more than one parent.



Entity-Relationship Model

Entity-Relationship Model or simply ER Model is a high-level data model diagram. In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand. It is also very easy for the developers to understand the system by just looking at the ER diagram

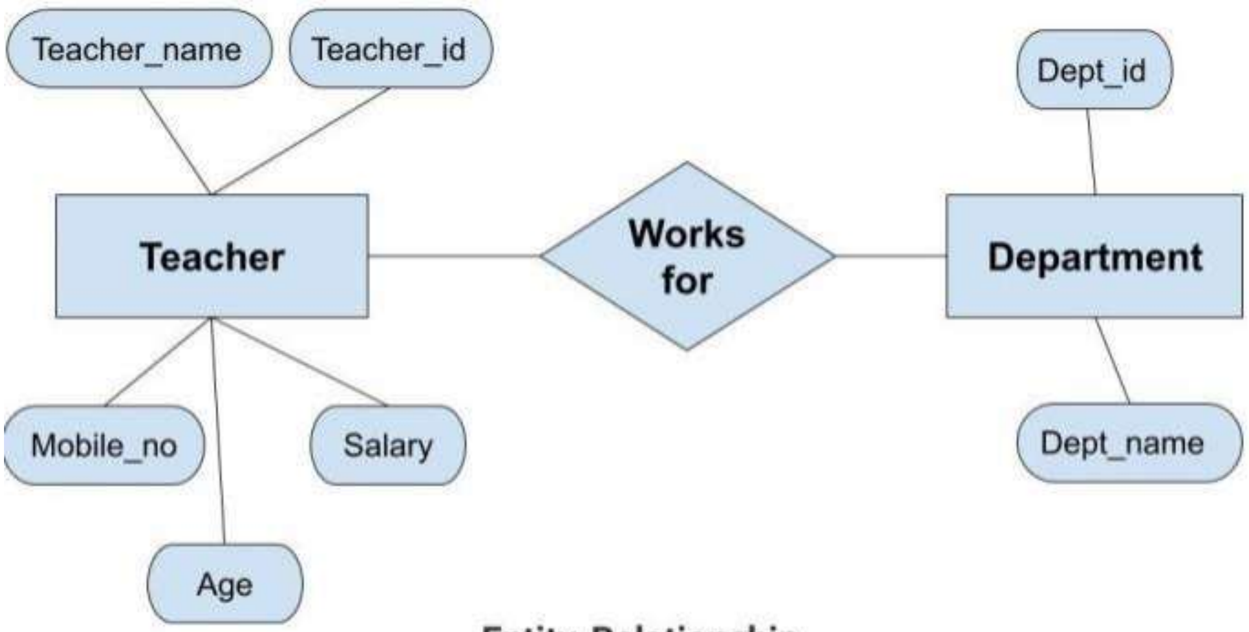
ER diagram has the following three components:

•**Entities:** Entity is a real-world thing. It can be a person, place, or even a concept. Example: Teachers, Students, Course, Building, Department, etc are some of the entities of a School Management System.

•**Attributes:** An entity contains a real-world property called attribute. This is the characteristics of that attribute. Example: The entity teacher has the property like teacher id, salary, age, etc.

•**Relationship:** Relationship tells how two attributes are related. Example: Teacher works for a department.

Example:

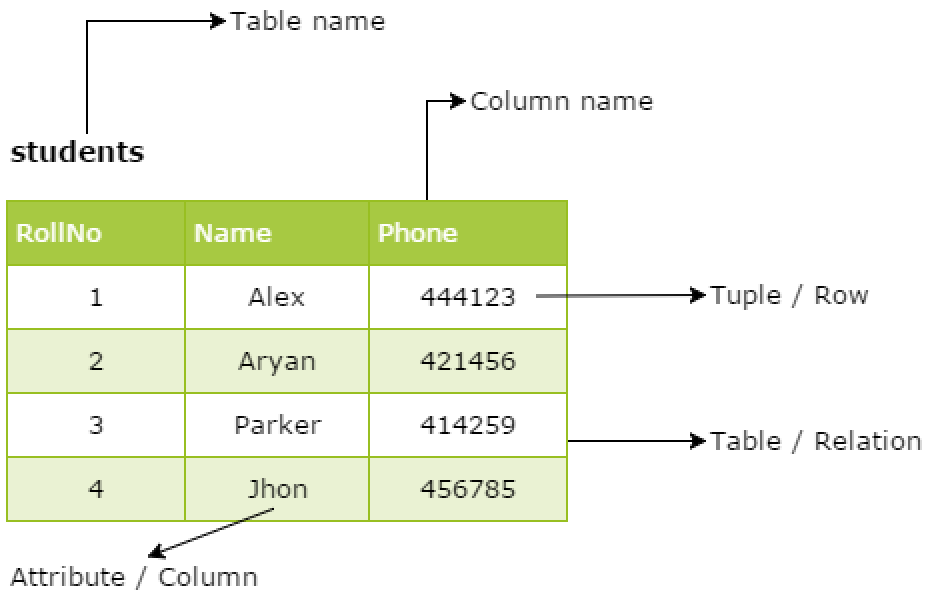


Entity-Relationship Model

In the above diagram, the entities are Teacher and Department. The attributes of Teacher entity are Teacher_Name, Teacher_id, Age, Salary, Mobile_Number. The attributes of entity Department entity are Dept_id, Dept_name. The two entities are connected using the relationship. Here, each teacher works for a department.

Relational Model

Relational Model is the most widely used model. In this model, the data is maintained in the form of a table. All the information is stored in the form of row and columns.
Tuples: Each row in the table is called tuple.
Attribute or Name: Each column in the table is called the attributes or the names. Attributes are the property which defines the table or relation.



RollNo	Name	Phone
1	Alex	444123
2	Aryan	421456
3	Parker	414259
4	Jhon	456785

Relational Model Terms

Object-based Data Model: An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types.

UNIT - 2

Relational data model:

•The relational model was introduced by Dr. E. F. Codd in 1970. It is a conceptual hay. of data based omit mathematical theory of relations.

•In a relational model, data is stored in tables. Three key terms are used frequently in relational database models: relations, attributes and domains,tuples

•A relation is a table with columns and rows. The named columns of the relation are called attributes. The domain is the set of values of the attributes. Rows are known as tuples.

•The relational models allow users to view the data logically rather than physically. The logical view of the relational database by the creation of data relationships through a table.

•A table is a two dimensional structure that contains rows & columns. The data is stored in these rows and columns. A table is also called relation.

For example: student table shown as

student

HTNO	STNAME	MARKS
101	AAA	90
102	BBB	20
103	CCC	80
104	DDD	90

Characteristics of a relation :

1. A table is viewed as a two dimensional structure that contains rows & columns.
2. Each attribute (or) column within a table has a unique name.
3. Only one value must be exist at the intersection of each row & column
4. Each row in a table should be unique.
5. Each table must have an attribute
6. each table have a primary key.
7. All values in a column must follow to the same data type.

E.F CODD RULES:

Rule 0: The Foundation Rule

The database must be in relational form. So that the system can handle the database through its relational capabilities.

Rule 1: Information Rule

A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.

Rule 2: Guaranteed Access Rule

Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.**Rule 3: Systematic Treatment of Null Values**

This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing the data, no value in a cell, inappropriate

information, unknown data and the primary key should not be null.
Rule 4: Active/Dynamic Online Catalog based on the relational model
It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database.
Rule 5: Comprehensive Data SubLanguage Rule
The relational database supports various languages, and if we want to access the database, the language must be the explicit, linear or well-defined syntax, character strings and supports the comprehensive: data definition, view definition, data manipulation, integrity constraints, and limit transaction management operations. If the database allows access to the data without any language, it is considered a violation of the database.
Rule 6: View Updating Rule
All views table can be theoretically updated and must be practically updated by the database systems.
Rule 7: Relational Level Operation (High-Level Insert, Update and delete) Rule
A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system.
Rule 8: Physical Data Independence Rule
All stored data in a database or an application must be physically independent to access the database. Each data should not depend on other data or an application. If data is updated or the physical structure of the database is changed, it will not show any effect on external applications that are accessing the data from the database.
Rule 9: Logical Data Independence Rule
It is similar to physical data independence. It means, if any changes occurred to the logical level (table structures), it should not affect the user's view (application). For example, suppose a table either split into two tables, or two table joins to create a single table, these changes should not be impacted on the user view application.
Rule 10: Integrity Independence Rule
A database must maintain integrity independence when inserting data into table's cells using the SQL query language. All entered values should not be changed or rely on any external factor or application to maintain integrity. It is also helpful in making the database independent for each front-end application.
Rule 11: Distribution Independence Rule
The distribution independence rule represents a database that must work properly, even if it is stored in different locations and used by different end-users. Suppose a user accesses the database through an application; in that case, they should not be aware that another user uses particular data, and the data they always get is only located on one site. The end users can access the database, and these access data should be independent for every user to perform the SQL queries.
Rule 12: Non Subversion Rule
The non-submersion rule defines RDBMS as a SQL language to store and manipulate the data in the database. If a system has a low-level or separate language other than SQL to access the database system, it should not subvert or bypass integrity to transform data.

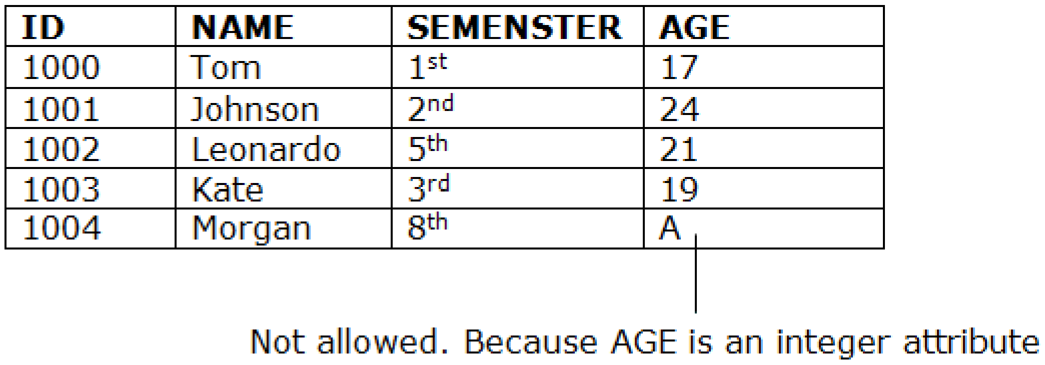
Integrity Constraints

o Integrity constraints are a set of rules. It is used to maintain the quality of information.
o Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
Types are

1. Domain constraints

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

Example:



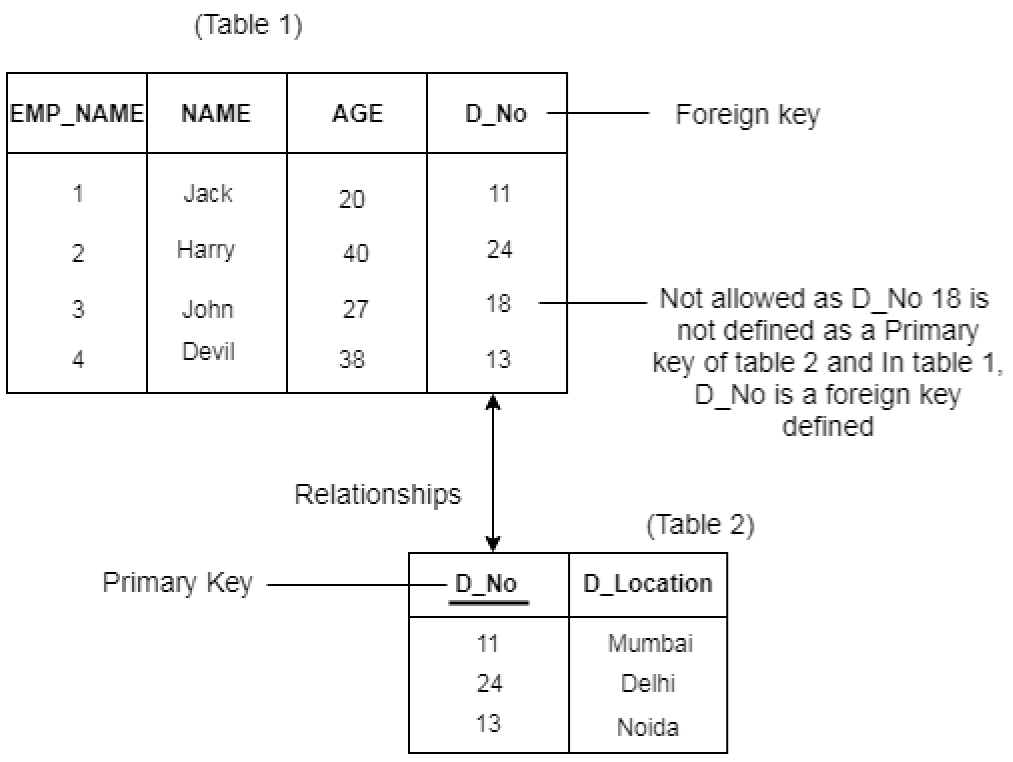
ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

Referential Integrity Constraints

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

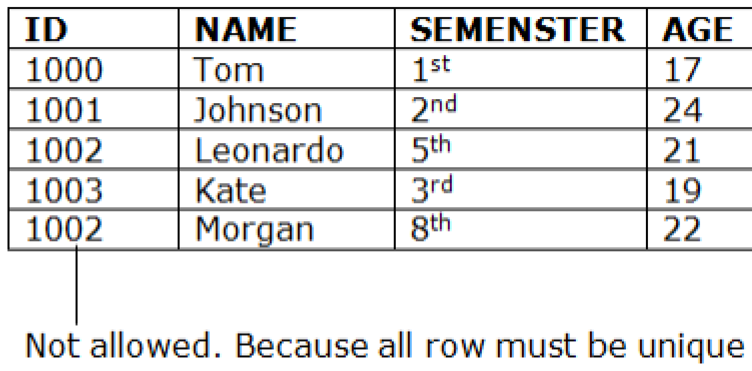
Example:


(Table 1)
Foreign key
Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined
Relationships
(Table 2)
Primary Key

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

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

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



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

Not allowed. Because all row must be unique

<u>KEYS in DBMS :</u>

•KEYS in DBMS is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table).
•They allow you to find the relation between two tables. Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.
•Key is also helpful for finding unique record or row from the table. Database key is also helpful for finding unique record or row from the table.

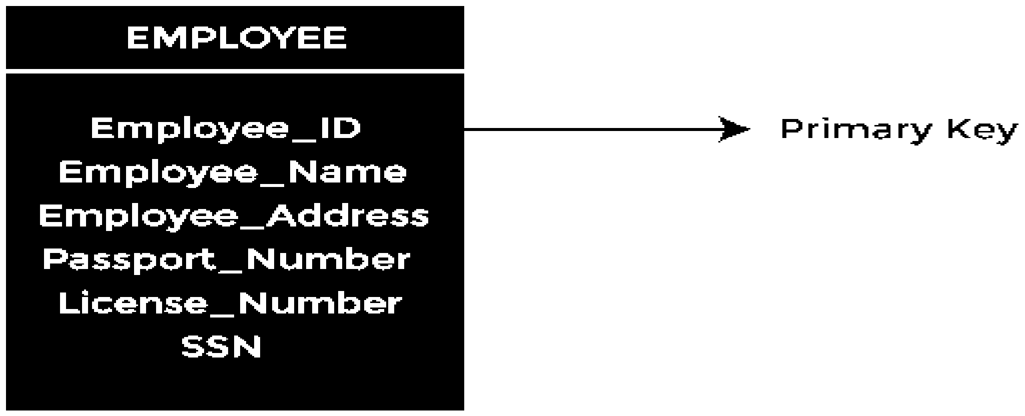
Types of Keys in DBMS

Primary key:

o It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.

o In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.

o For each entity, the primary key selection is based on requirements and developers



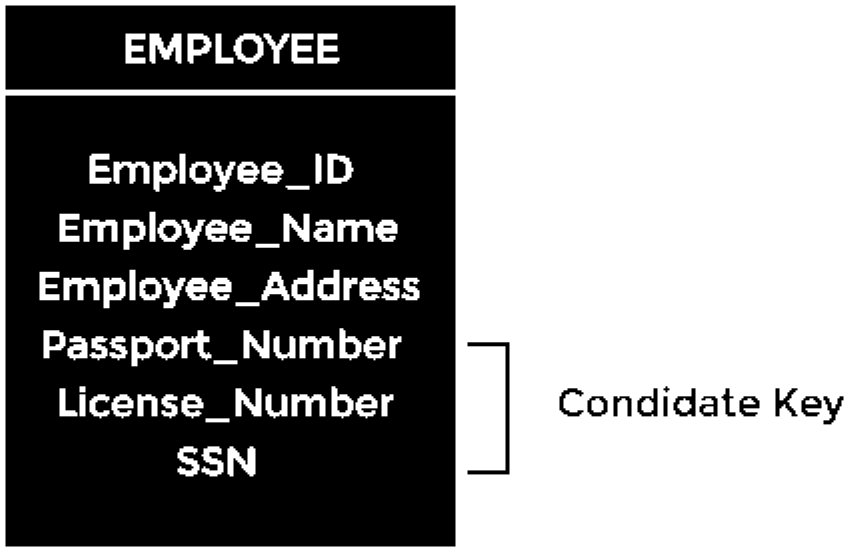
2. Candidate key

o A candidate key is an attribute or set of attributes that can uniquely identify a tuple.

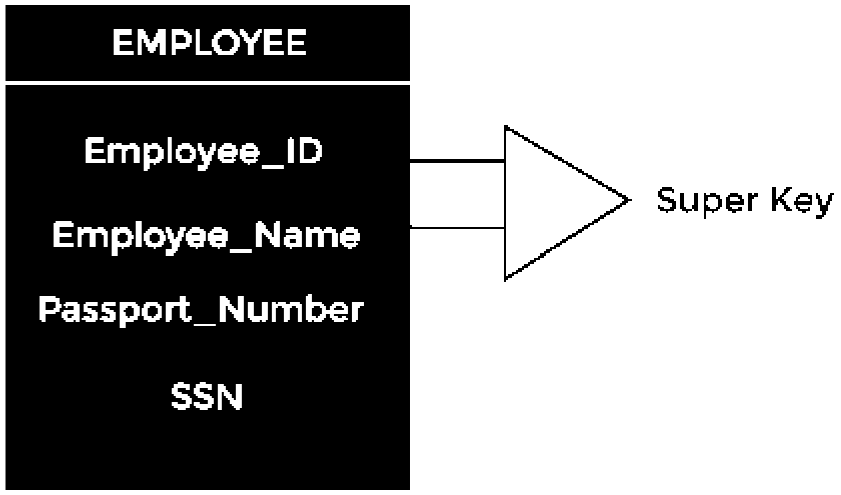
o Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

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

attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Super Key
Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a Primary key and non key attribute. Ex: { employee_id,employee_name}

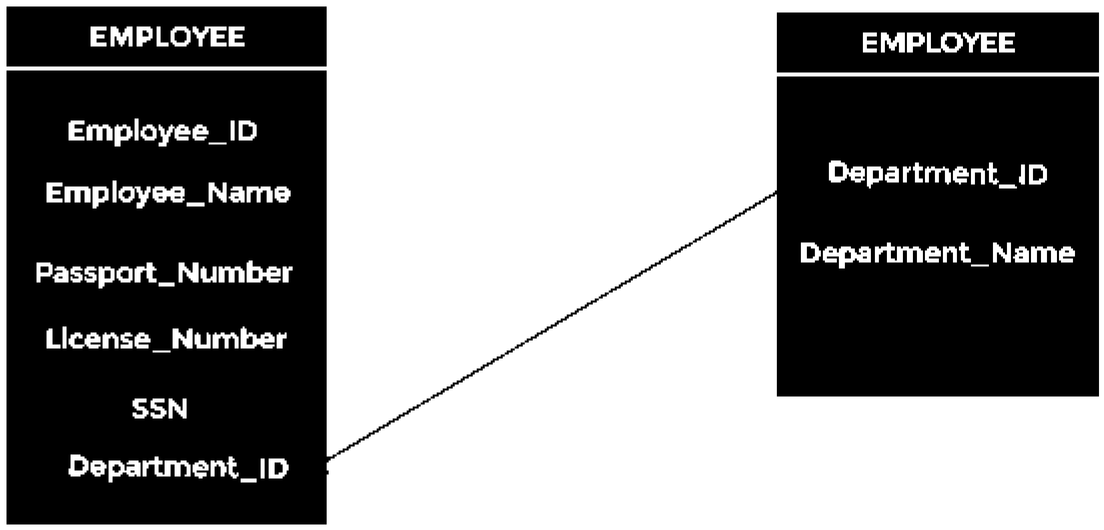


For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

4. Foreign key

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

- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



Normalization :

It is a process of dividing larger tables into smaller tables

•Normalization is the process of organizing the data and the attributes of a database. It is performed to reduce the data redundancy in a database and to ensure that data is stored logically.

•Data redundancy in DBMS means having the same data but at multiple places.

•Normalization is implemented by using Normal forms.

Normal Forms

First Normal Form (1NF)

A relation is in 1NF if every attribute is a single-valued attribute or it does not contain any multi-valued or composite attribute, i.e., every attribute is an atomic attribute. If there is a composite or multi-valued attribute, it violates the 1NF.

Let’s take an example of a relational table employee table that contains the details of the employees of the company.

Employee

Employee Code	Employee Name	Employee Phone Number
101	John	98765623,998234123
101	John	89023467
102	Ryan	76213908
103	Stephanie	98132452

Here, the Employee Phone Number is a multi-valued attribute. So, this relation is not in 1NF. To convert this table into 1NF, we make new rows with each Employee Phone Number as a new row as shown below:

Employee

Employee Code	Employee Name	Employee Phone Number
101	John	998234123
101	John	98765623
101	John	89023467
102	Ryan	76213908
103	Stephanie	98132452

Second Normal Form (2NF)

The normalization of 1NF relations to 2NF involves the elimination of partial dependencies.

Partial functional dependency:

A partial dependency in DBMS exists when any non-prime attributes, i.e., an attribute not a part of the candidate key, is not fully functionally dependent on one of the candidate keys.

For a relational table to be in second normal form, it must satisfy the following rules:

1. The table must be in first normal form.
2. It must not contain any partial dependency, i.e., all non-prime attributes are fully functionally dependent on the primary key

If a partial dependency exists, we can divide the table to remove the partially dependent attributes and move them to some other table where they fit in well.

Let us take an example of the following EmployeeProjectDetail table to understand what is partial dependency and how to normalize the table to the second normal form:

EmployeeProjectDetail

Employee Code	Project ID	Employee Name	Project Name
101	P03	John	Project 103
101	P01	John	Project 101
102	P04	Ryan	Project104
103	P02	Stephanie	Project102

In the above table, the prime attributes of the table are Employee Code and Project ID. We have partial dependencies in this table because Employee Name can be determined by Employee Code and Project Name can be determined by Project ID. Thus, the above relational table violates the rule of 2NF.

The prime attributes in DBMS are those which are part of one or more primary keys.

To remove partial dependencies from this table and normalize it into second normal form, we can decompose the EmployeeProjectDetail table into the following three tables:

EmployeeDetail

Employee Code	Employee Name
101	John
101	John
102	Ryan
103	Stephanie

ProjectDetail

Project ID	Project Name
P03	Project 103
P01	Project 101
P04	Project104
P02	Project102

Third Normal Form (3NF)

The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies in DBMS.

Transitive dependency:

It defines there is a dependency between two non key attributes in a table.

For a relational table to be in third normal form, it must satisfy the following rules:

1. The table must be in the second normal form.
2. No non-key attribute is transitively dependent on the primary key.

If a transitive dependency exists, we can divide the table to remove the transitively dependent attributes and place them to a new table along with a copy of the determinant.

Let us take an example of the following <collegeDetail> table to understand what is transitive dependency and how to normalize the table to the third normal form:

<college Detail>

college Code	college Name	Principal name
101	ABC	JOHN
102	Def	RAVI
103	MNO	RAJU
104	PQR	RANI

The above table is not in 3NF because it has college name -> principal name transitive dependency

To remove transitive dependency from this table and normalize it into the third normal form, we can decompose the <collegeDetail> table into the following two tables:

<college>

college Code	college Name
101	ABC
102	Def
103	MNO
104	PQR

<principal>

college Code	Principal name
101	JOHN
102	RAVI
103	RAJU
104	RANI

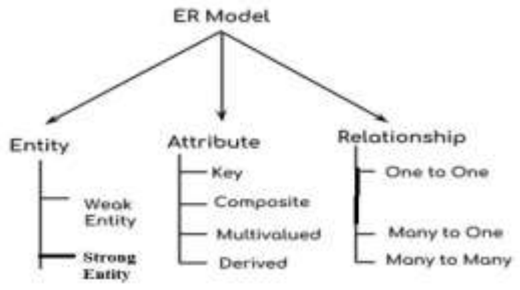
Boyce codd normal form:
It is a data base normalization technique that ensures data consistency and reduces data redundancy. It is an extension of 3NF and is considered as higher level normalization.

UNIT-3

Entity–Relationship Model (Er Model)

•An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram).
•An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are Entities, Attributes and Relationships
•An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

Components of a ER Diagram /Basic building blocks of an ER diagram:



As shown in the above diagram, an ER diagram has three main components:
1. Entity
2. Attribute
3. Relationship

1. Entity

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

1. Entity

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

Classification Of Entity Set:

An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; Number of students in a class or college.

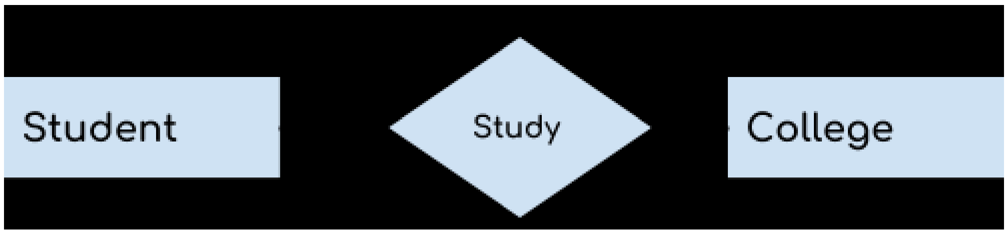
Entity set can be classified into two types

1. Strong Entity and
2. Weak Entity

1. Strong Entity:

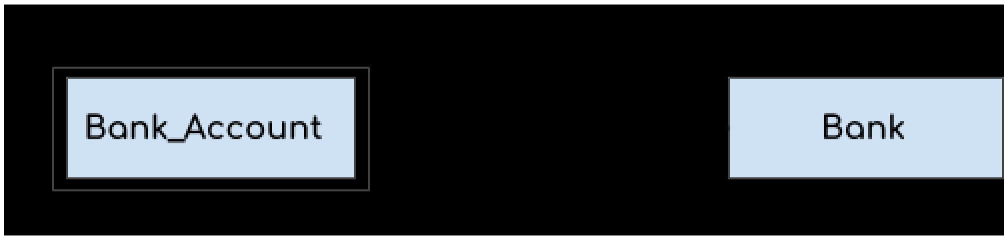
An entity is represented as rectangle in an ER diagram.

For example: In the following ER diagram we have two entities Student and College and these two entities have many to one relationship as many students study in a single college.



Weak Entity:

An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle. For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity



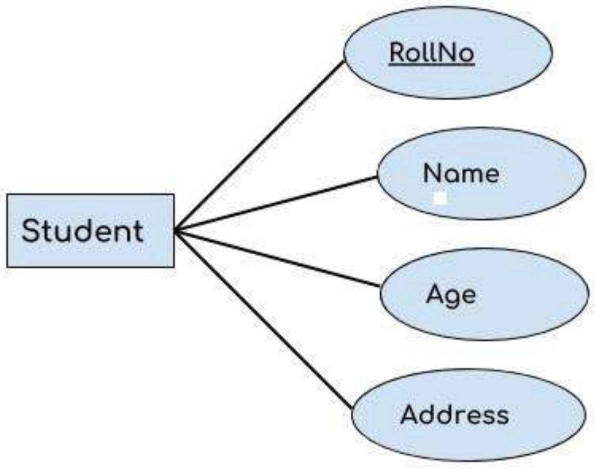
2. Attribute

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

1. Key attribute

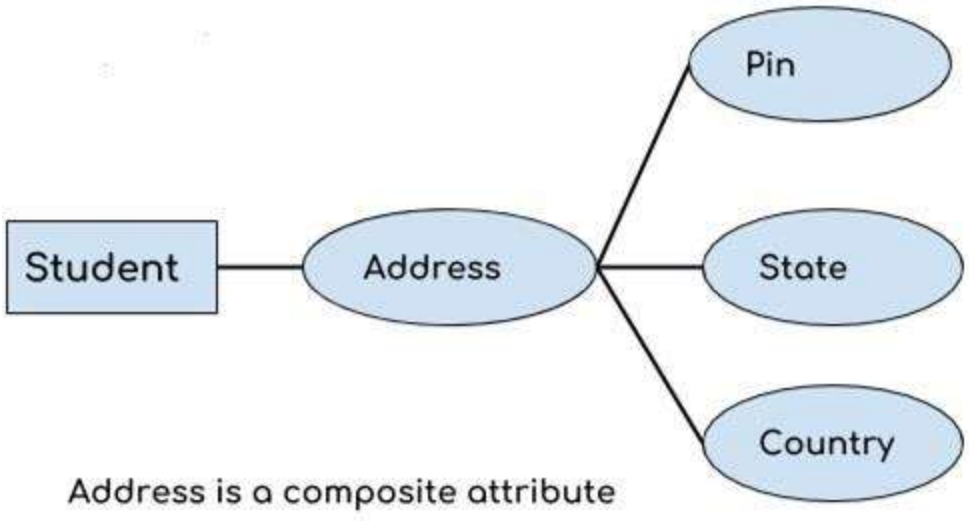
2. Composite attribute
3. Multivalued attribute
4. Derived attribute

1. Key attribute:



A key attribute can uniquely identify an entity from an entity set. For example, in the above diagram student roll number can uniquely identify a student from a set of students. Key attribute is represented by oval same as other attributes however the text of key attribute is underlined.

2. Composite attribute:



An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.

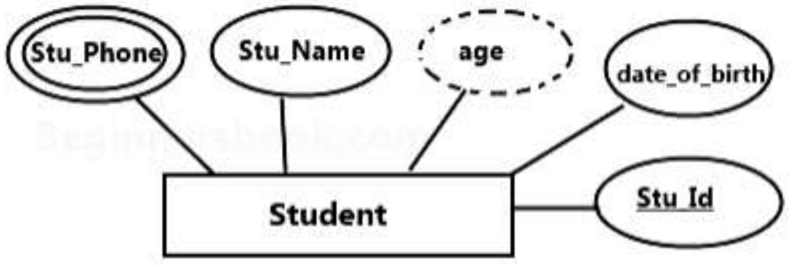
3. Multivalued attribute:

An attribute that can hold multiple values is known as multivalued attribute. It is represented with double ovals in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

4. Derived attribute:

A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by dashed oval in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).

E-R diagram with multivalued and derived attributes:



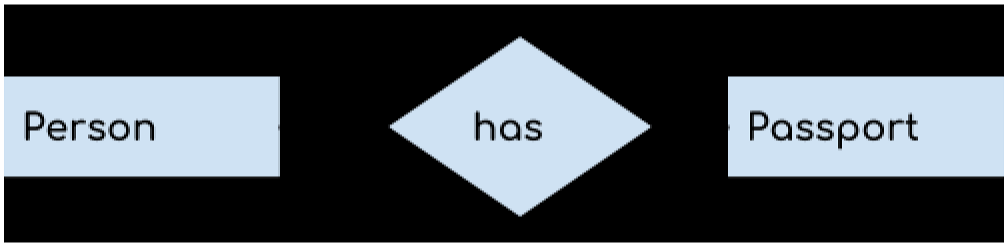
3. Relationship

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:

1. One to One
2. One to Many

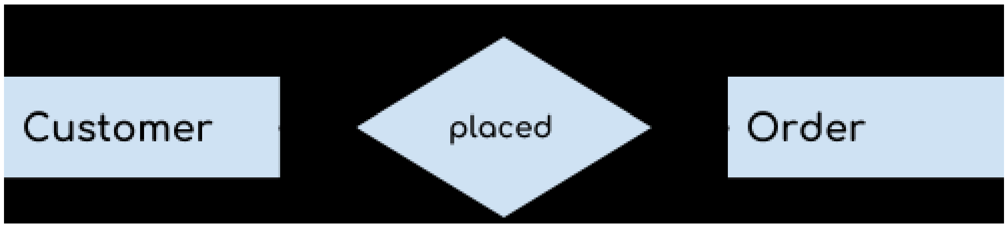
3. Many to Many

1. One to One Relationship



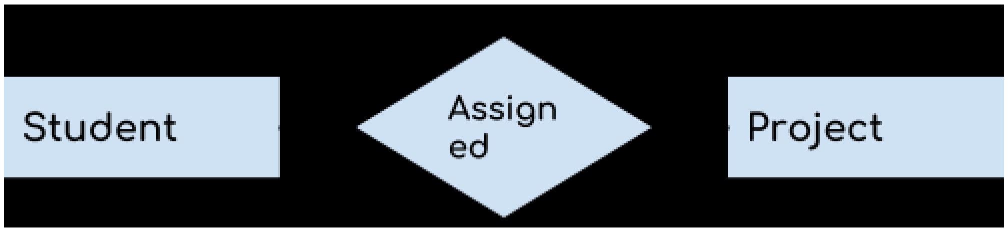
When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person.

2. One to Many Relationship



When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship. For example – a customer can place many orders but a order cannot be placed by many customers.

3. Many to Many Relationship



When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship. For example, a can be assigned to many projects and a project can be assigned to many students.

<u>Generalization And Specialization</u>

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:

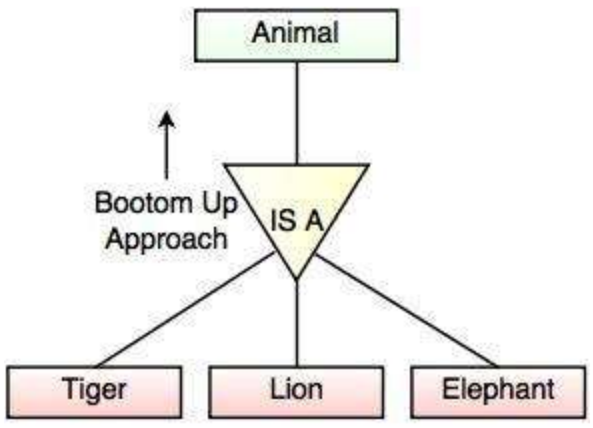


Fig. Generalization

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

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.

For example

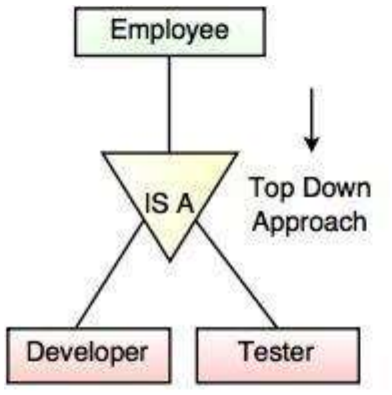


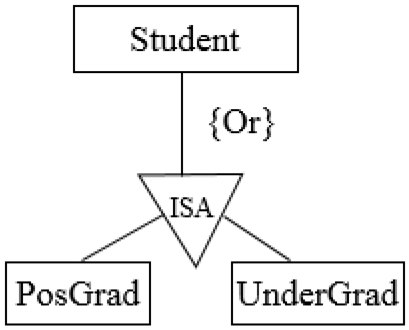
Fig. Specialization

In the above example, Employee can be specialized as Developer or Tester, based on what role they play in an Organization.

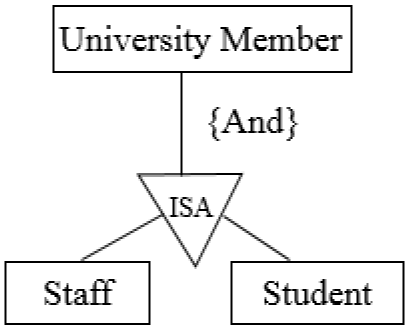
Constraints On Specialization/Generalization

There are three constraints that may apply to a specialization/generalization:

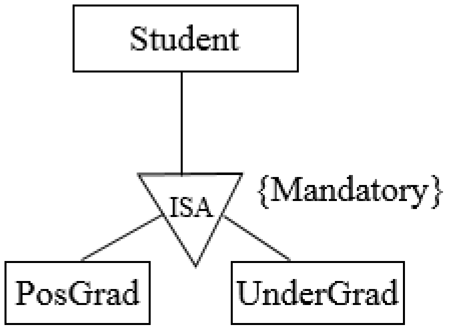
Disjoint: The disjoint constraint only applies when a superclass has more than one subclass. If the subclasses are disjoint, then an entity occurrence can be a member of only one of the subclasses, e.g. postgrads or undergrads ,you cannot be both.



Overlapping: This applies when an entity occurrence may be a member of more than one subclass, e.g. student and staff some people are both.



Completeness constraints
Total: Each superclass (higher-level entity) must belong to subclasses (lower-level entity sets), e.g. a student must be postgrad or undergrad.



BASIC SQL

o SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDMS).
o It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
o All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.
o SQL allows users to query the database in a number of ways, using English-like statements.

SQL Datatype

o SQL Datatype is used to define the values that a column can contain.
o Every column is required to have a name and data type in the database table

Int	It is used to specify an integer value
Numeric	It is used to specify a numeric value
char	It has a maximum length of 8000 characters. It contains Fixed-length non unicode characters.
varchar	It has a maximum length of 8000 characters. It contains variable-length non unicode characters.
Date	It is used to store the year, month, and days value.
time	It is used to store the hour, minute, and second values.

SQL Commands

o SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
o SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

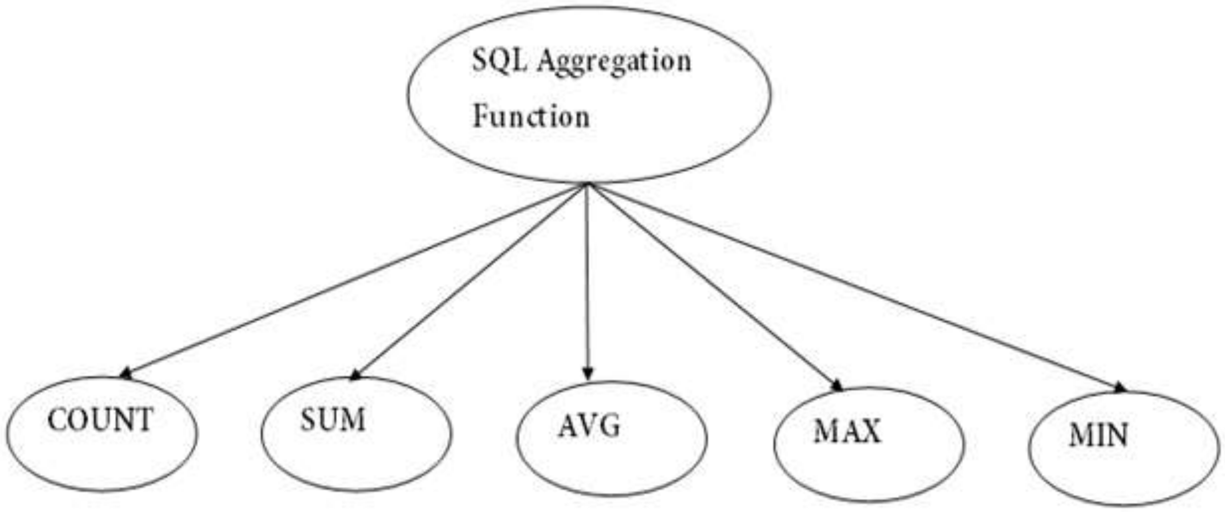
Types Of Sql Commands

1. Data Definition Language (DDL)
o DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
o All the command of DDL are auto-committed that means it permanently save all the changes in the database.
Here are some commands that come under DDL:
o CREATE
o ALTER
o DROP
o TRUNCATE
CREATE It is used to create a new table in the database.
Syntax:
CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);
Example:
CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100));
DROP: It is used to delete both the structure and record stored in the table.
Syntax
DROP TABLE table_name;
Example
DROP TABLE EMPLOYEE;
ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.
Syntax:
To add a new column in the table

ALTER TABLE table_name ADD column_name COLUMN-definition;
EXAMPLE
ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));
TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.
Syntax:
TRUNCATE TABLE table_name;
Example:
TRUNCATE TABLE EMPLOYEE;
2. DATA MANIPULATION LANGUAGE
o DML commands are used to modify the database. It is responsible for all form of changes in the database.
o The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.
Here are some commands that come under DML:
o INSERT
o UPDATE
o DELETE
a**. INSERT**: The INSERT statement is a SQL query. It is used to insert data into the row of a table.
Syntax:
1. INSERT INTO TABLE_NAME
(col1, col2, col3,.... col N)
VALUES (value1, value2, value3, ... valueN);
Or
1. INSERT INTO TABLE_NAME
2. VALUES (value1, value2, value3, ... valueN);
For example:
Insert into student values(1,’sony’);
b. UPDATE: This command is used to update or modify the value of a column in the table.
Syntax:
UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]
For example:
UPDATE students SET User_Name = 'Sonoo' WHERE Student_Id = '3'
c. DELETE: It is used to remove one or more row from a table.
Syntax:
DELETE FROM table_name [WHERE condition];
For example:
DELETE FROM student WHERE name="Sonoo";

Aggregate Functions

o SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.
o It is also used to summarize the data.
Types of SQL Aggregation Function



1**. COUNT FUNCTION**
o COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.
o COUNT function uses the COUNT(*) that returns the count of all the rows in a specified table. COUNT(*) considers duplicate and Null.
Syntax
COUNT(*)
Ex: 1. PRODUCT

PRODUCT	COMPANY	QTY	RATE	COST
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60
Item4	Com3	5	10	50
Item5	Com2	2	20	40
Item6	Com1	3	25	75
Item7	Com1	5	30	150
Item8	Com1	3	10	30
Item9	Com2	2	25	50

Example: COUNT()
SELECT COUNT(*) FROM PRODUCT;
Output:
09

Example: COUNT with WHERE
SELECT COUNT(*) FROM PRODUCT WHERE RATE>=20;
Output:
6

SUM Function

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

Syntax
SUM()

example: SUM()
SELECT SUM(COST) FROM PRODUCT;

Output:
550

AVG function

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

Syntax
AVG()

Example:
SELECT AVG(COST) FROM PRODUCT;

Output:
61.1

MAX Function

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

Syntax
MAX()

Example:
SELECT MAX(RATE) FROM PRODUCT;
30

MIN Function

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

Syntax
MIN()

Example:
SELECT MIN(RATE) FROM PRODUCT;

Output:
10

Grouping and ordering:

The GROUP BY statement groups rows that have the same values into summary rows.
The GROUP BY statement is often used with aggregate functions
(COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

Ex:
Select name,max(qty) from product group by company;

The ORDER BY keyword is used to sort the result-set in ascending or descending order.
Ex:
SELECT * FROM Products
ORDER BY Price;

SQL Operators

Operator is a symbol used to perform operation.

SQL Arithmetic Operators

The **Arithmetic Operators** perform the mathematical operation on the numerical data of the SQL tables. These operators perform addition, subtraction, multiplication, and division operations on the numerical operands.
Following are the various arithmetic operators performed on the SQL data:

1. SQL Addition Operator (+)
2. SQL Subtraction Operator (-)
3. SQL Multiplication Operator (*)
4. SQL Division Operator (/)

SQL Addition Operator (+)

The **Addition Operator** in SQL performs the addition on the numerical data of the database table. In SQL, we can easily add the numerical values of two columns of the same table by specifying both the column names as the first and second operand.
Syntax:

SELECT operand1 + operand2;

SQL Subtraction Operator (-)

The Subtraction Operator in SQL performs the subtraction on the numerical data of the database table. In SQL, we can easily subtract the numerical values of two columns of the same table by specifying both the column names as the first and second operand.
Syntax:

SELECT operand1 - operand2;

SQL Multiplication Operator (*)

The Multiplication Operator in SQL performs the Multiplication on the numerical data of the database table. In SQL, we can easily multiply the numerical values of two columns of the same table by specifying both the column names as the first and second operand.

Syntax:

SELECT operand1 * operand2;

SQL Division Operator (/)

The Division Operator in SQL divides the operand on the left side by the operand on the right side.

Syntax:

SELECT operand1 / operand2;

SQL Logical Operators

Operator	**Description**
AND	TRUE if all the conditions separated by AND is TRUE
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE