



**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108

Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



---

## Department of CSE (Data Science)

---

### DBMS

### UNIT-I

## Introduction:-

**Database Management System (DBMS)** is software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs that manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software store and retrieve data.

DBMS allows users to create their own databases as per their requirements. The term "DBMS" includes the user of the [database](#) and other application programs. It provides an interface between the data and the software application.

In this Database Management System tutorial, you will learn DBMS concepts like-

## Example of a DBMS

Let us see a simple example of a university database. This database is maintaining information concerning students, courses, and grades in a university environment. The database is organized as five files:

- The STUDENT file stores the data of each student
- The COURSE file stores contain data on each course.
- The SECTION stores information about sections in a particular course.
- The GRADE file stores the grades which students receive in the various sections
- The TUTOR file contains information about each professor.

To define DBMS:

- We need to specify the structure of the records of each file by defining the different types of data elements to be stored in each record.
- We can also use a coding scheme to represent the values of a data item.
- Basically, your Database will have 5 tables with a foreign key defined amongst the various tables.

## Advantages of DBMS



---

### Department of CSE (Data Science)

---

- DBMS offers a variety of techniques to store & retrieve data
- DBMS serves as an efficient handler to balance the needs of multiple applications using the same data
- Uniform administration procedures for data
- Application programmers are never exposed to details of data representation and storage.
- A DBMS uses various powerful functions to store and retrieve data efficiently.
- Offers Data Integrity and Security
- The DBMS implies integrity constraints to get a high level of protection against prohibited access to data.
- A DBMS schedules concurrent access to the data in such a manner that only one user can access the same data at a time
- Reduced Application Development Time

## Disadvantage of DBMS

DBMS may offer plenty of advantages, but it has certain flaws-

- The cost of Hardware and Software of a DBMS is quite high, which increases the budget of your organization.
- Most database management systems are often complex, so training users to use the DBMS is required.
- In some organizations, all data is integrated into a single database that can be damaged because of electric failure or corruption in the storage media.
- Using the same program at a time by multiple users sometimes leads to data loss.
- DBMS can't perform sophisticated calculations

## Significance

Most DBMS include the following:

- **Storage engine** - As the core component of a DBMS, this stores the data. It's the part of the system that communicates with the file system at the OS level. It's the gateway for all the SQL queries that interact with the stored data.
- **System catalog or database dictionary** - Also called the metadata catalog, this component is a centralized repository for all created database objects. It is used to confirm data requests from users and also to provide information about a database's objects, security, performance, and more.
- **Database access language** - every DBMS needs an application programming interface (API) to enable users to create databases and access data, and it usually comes in the form



---

### Department of CSE (Data Science)

---

of a database access language. For instance, structured query language (SQL) is the default data access language in relational databases.

- **Optimization engine** - This component processes data requests and transforms them into actionable commands. It also helps tune databases for optimal performance.
- **Query processor** - Once a query (data request) has gone through the optimization engine, the query processor handles the request and feeds back the results. It acts as a sort of middleman between the database and user queries.
- **Lock manager** - This component keeps multiple users from modifying the same data at the same time. It locks access for each user in turn.
- **Log manager** - All DBMS keep records of how and when data in the database is modified, created, or deleted. The log manager records this information and can also integrate with database utilities to recover data or make backups. It manages the logs by organizing them and keeping them easily accessible.
- **Data utilities** - This category is an umbrella term for a variety of components that simplify database management and monitor activity. They can include software for data backup and restore, integrity checks, reporting and monitoring, simple repair, validations, and so on.

## Types of DBMS



### Types of DBMS

The main Four Types of Database Management Systems are:

- Hierarchical database
- Network database
- Relational database
- Object-Oriented database



---

## Department of CSE (Data Science)

---

### Hierarchical DBMS

In a Hierarchical database, model data is organized in a tree-like structure. Data is Stored Hierarchically (top-down or bottom-up) format. Data is represented using a parent-child relationship. In Hierarchical DBMS, parents may have many children, but children have only one parent.

### Network Model

The network database model allows each child to have multiple parents. It helps you to address the need to model more complex relationships like the orders/parts many-to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths.

### Relational Model

Relational DBMS is the most widely used DBMS model because it is one of the easiest. This model is based on normalizing data in the rows and columns of the tables. Relational model stored in fixed structures and manipulated using SQL.

### Object-Oriented Model

In the Object-oriented Model data is stored in the form of objects. The structure is called classes which display data within it. It is one of the components of DBMS that defines a database as a collection of objects that stores both data members' values and operations.

## DBMS Architecture

A **Database Architecture** is a representation of DBMS design. It helps to design, develop, implement, and maintain the database management system. A DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered. It also helps to understand the components of a database.

A [Database](#) stores critical information and helps access data quickly and securely. Therefore, selecting the correct Architecture of DBMS helps in easy and efficient data management.

## Types of DBMS Architecture

There are mainly three types of DBMS architecture:

- One Tier Architecture (Single Tier Architecture)
- Two Tier Architecture



---

**Department of CSE (Data Science)**

---

- Three Tier Architecture

Now, we will learn about different architecture of DBMS with diagram.

## 1-Tier Architecture

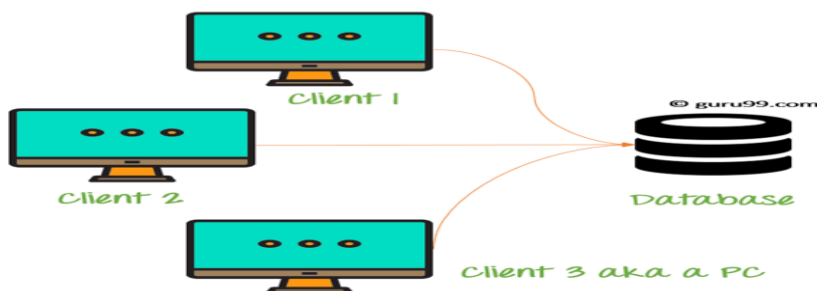
**Tier Architecture** in DBMS is the simplest architecture of Database in which the client, server, and Database all reside on the same machine. A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.



**1 Tier Architecture Diagram**

## 2-Tier Architecture

A **2 Tier Architecture** in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the second tier. Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.



**2 Tier Architecture Diagram**



---

## Department of CSE (Data Science)

---

In the above 2 Tier client-server architecture of database management system, we can see that one server is connected with clients 1, 2, and 3.

### Two Tier Architecture Example:

A Contact Management System created using MS- Access.

### 3-Tier Architecture

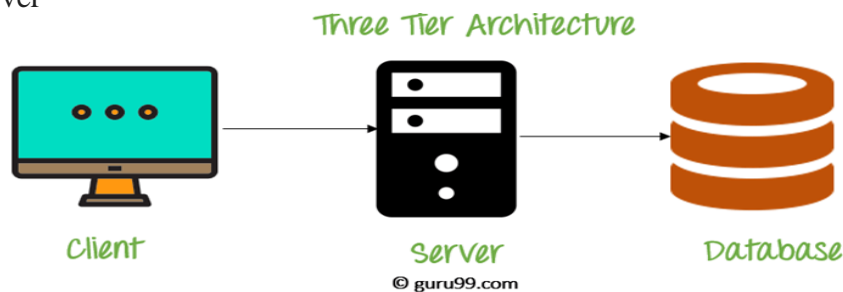
A **3 Tier Architecture** in DBMS is the most popular client server architecture in DBMS in which the development and maintenance of functional processes, logic, data access, data storage, and user interface is done independently as separate modules. Three Tier architecture contains a presentation layer, an application layer, and a database server.

3-Tier database Architecture design is an extension of the 2-tier client-server architecture. A 3-tier architecture has the following layers:

Presentation layer (your PC, Tablet, Mobile, etc.)

Application layer (server)

Database Server



**3 Tier Architecture Diagram**

The Application layer resides between the user and the DBMS, which is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user. The application layer(business logic layer) also processes functional logic, constraint, and rules before passing data to the user or down to the DBMS.

### The goal of Three Tier client-server architecture is:

To separate the user applications and physical database

To support DBMS characteristics

Program-data independence

Supporting multiple views of the data



---

## Department of CSE (Data Science)

---

## Function of DBMS

### Data Storage Management

One of the most important tasks for DBMS is to create a database for complex data and manage the data. It gives relief to the user by creating a structure for the complex data sets so that users can access it and manipulate them very easily. Modern database systems not only provide storage for the data but they store and manage the metadata (data of data) like data procedural rules, validation rules etc. DBMS also provides performance tuning, which makes accessing data faster and easier.

### Security Management

Security is another aspect which is handled by the Database Management Systems. Database systems provide a high level of security measures using various security algorithms to keep the data safe and ensure the data privacy. There are certain security rules that ensure what data can be accessed from the database and which user can access it. It also makes sure what operations (read, write, delete) can be performed on the specific data. It is very important for the organizations where multi-user databases are required.

### Backup and Recovery Management

To keep the data safe and ensure the integrity, the database system provides the features for backup and recovery management. If the system fails due to some reason then it recovers the data and keeps the data safe.

### Database Access Language and Application Programming Interface

DBMS provides a database access language which is also called a query language. Query languages are non-procedural languages used to access the database and manipulate the data. SQL is an example of a query language. The majority of DBMS vendors provide the support of various query languages to access the data.

### Data Dictionary Management

Data dictionary management is also useful functionality provided by the Database Management System. In the data dictionary, it stores the data and its related information about its relationship. So, a data dictionary keeps the data structures and their relationships with other data, so that a



---

## Department of CSE (Data Science)

---

programmer is not responsible for storing the relationship in the database through complex coding. DBMS provides the data abstraction and removes the dependency of the data from the system.

## Data Transformation and Presentation

DBMS provides the functionality of data transformation, which means programmers need not worry about the logical and physical representation of the data. DBMS stores the data in the determined data structure.

For example, if a user asks for the date from a database and he receives it as 14 December 2022, but in the database, it is stored in different columns of month, date and year.

## Multi User Access Control

Multi User Access control is another feature which is provided by the modern Database Systems. So, more than one user can access the database at the same time without any problem. This feature makes sure the integrity of the data present in the database. It also follows the ACID property, so the database will be consistent while multiple users are accessing it concurrently. It is very useful for the database of organizations where multiple database engineers are working concurrently.

## Data Integrity Management

Database systems provide data integrity management by maximizing the data consistency and minimizing the data redundancy. The data dictionary is the feature database system used to store the relationships of the data to keep the data integrity. Data integrity is needed where a transaction based database system is present.

## Database Communication Interface

When a user requests data from the database, it uses some environments like browsers (Chrome or Firefox etc.) to get the data.

**An end user can access data in the following ways:**

- If he asks for the data through any form and sends the request.
- He can get the data if DBMS publishes the data on any website without asking him.
- He can get the data through a third party distribution network.





## Relational Model in DBMS

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

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

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

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

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

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

**Example: STUDENT Relation**

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

## Introduction of ER Model

The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.



---

**Department of CSE (Data Science)**

---

The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, the ER Diagram is the structural format of the database.







### Why Use ER Diagrams In DBMS?

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

### Symbols Used in ER Model

ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

- **Rectangles:** Rectangles represent Entities in the ER Model.
- **Ellipses:** Ellipses represent Attributes in the ER Model.
- **Diamond:** Diamonds represent Relationships among Entities.
- **Lines:** Lines represent attributes to entities and entity sets with other relationship types.
- **Double Ellipse:** Double Ellipses represent Multi-Valued Attributes.
- **Double Rectangle:** Double Rectangle represents a Weak Entity.

Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse		Attributes in ER Model
Diamond		Relationships among Entities
Line		Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse		Multi-Valued Attributes
Double Rectangle		Weak Entity

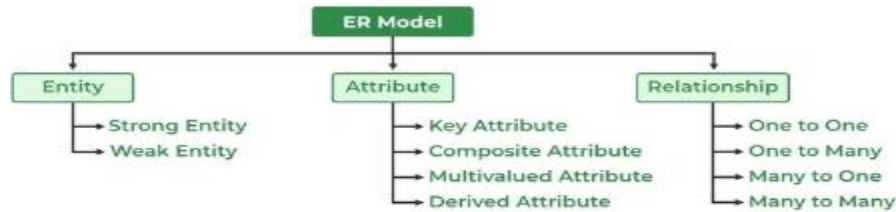
*Symbols used in ER Diagram*

### Components of ER Diagram

ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.



Department of CSE (Data Science)

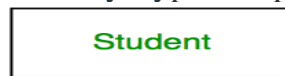


Components of ER Diagram

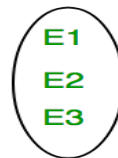
## Entity

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.

**Entity Set:** An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



Entity Type



Entity Set

### 1. Strong Entity

A **Strong Entity** is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

### 2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can't be defined. These are called **Weak Entity types**.

**For Example,** A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents don't have existed without the employee. So Dependent will be a **Weak Entity Type** and Employee will be Identifying Entity type for Dependent, which means it is **Strong Entity Type**.



## Department of CSE (Data Science)

A weak entity type is represented by a Double Rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



### Attributes

[Attributes](#) are the properties that define the entity type. For example, Roll\_No, Name, DOB, Age, Address, and Mobile\_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.



Attribute

#### 1. Key Attribute

The attribute which **uniquely identifies each entity** in the entity set is called the key attribute. For example, Roll\_No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines.



Key Attribute

#### 2. Composite Attribute

An attribute **composed of many other attributes** is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. In ER diagram, the composite attribute is represented by an oval comprising of ovals.



---

**Department of CSE (Data Science)**

---



### 3. Multivalued Attribute

An attribute consisting of more than one value for a given entity. For example, Phone\_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.



*Multivalued Attribute*

### 4. Derived Attribute

An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.

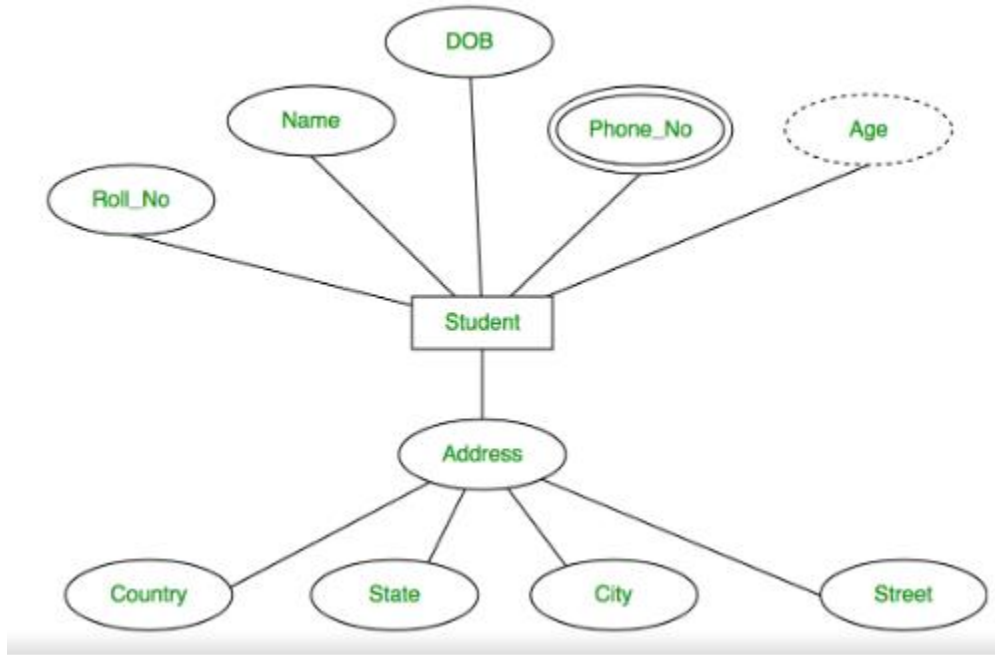


*Derived Attribute*

The Complete Entity Type Student with its Attributes can be represented as:



**Department of CSE (Data Science)**



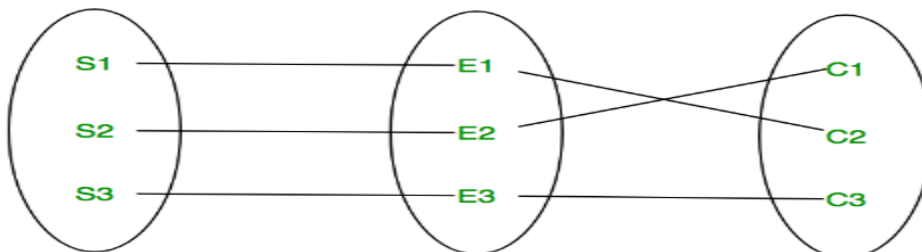
**Relationship Type and Relationship Set**

A Relationship Type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course. In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



*Entity-Relationship Set*

A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.



*Relationship Set*



---

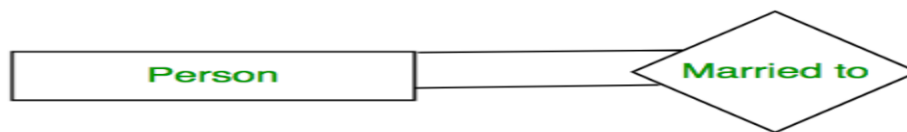
**Department of CSE (Data Science)**

---

*Degree of a Relationship Set*

The number of different entity sets participating in a relationship set is called the [degree of a relationship set](#).

**1. Unary Relationship:** When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



*Unary Relationship*

**2. Binary Relationship:** When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.



*Binary Relationship*

**3. n-ary Relationship:** When there are n entities set participating in a relation, the relationship is called an n-ary relationship.

*Cardinality*

The number of times an entity of an entity set participates in a relationship set is known as [cardinality](#). Cardinality can be of different types:

**1. One-to-One:** When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one. the total number of tables that can be used in this is 2.

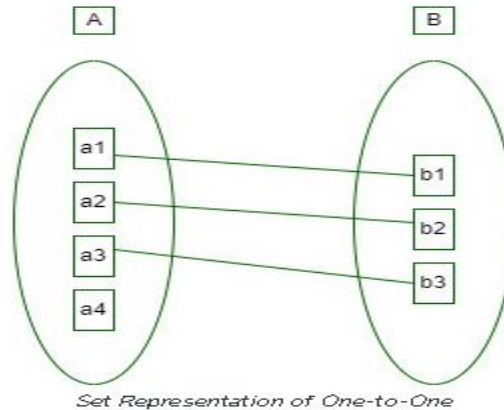


*one to one cardinality*

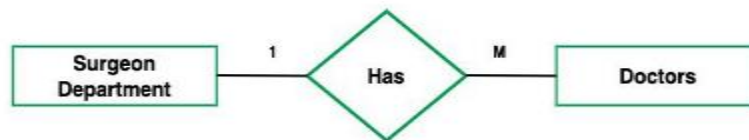
Using Sets, it can be represented as:



Department of CSE (Data Science)

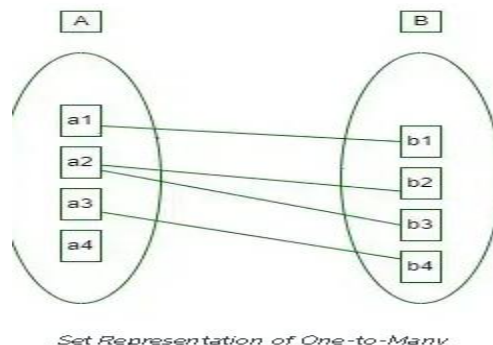


**2. One-to-Many:** In one-to-many mapping as well where each entity can be related to more than one relationship and the total number of tables that can be used in this is 2. Let us assume that one surgeon department can accommodate many doctors. So the Cardinality will be 1 to M. It means one department has many Doctors. total number of tables that can be used is 3.



one to many cardinality

Using sets, one-to-many cardinality can be represented as:



Set Representation of One-to-Many





Department of CSE (Data Science)

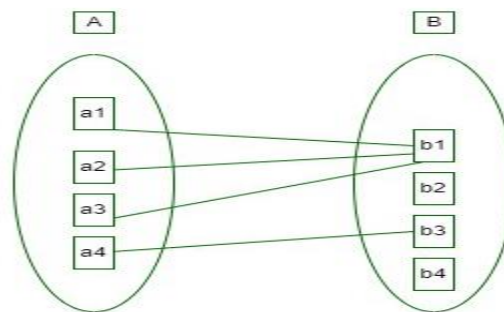
**3. Many-to-One:** When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be  $n$  to  $1$ . It means that for one course there can be  $n$  students but for one student, there will be only one course.

The total number of tables that can be used in this is 3.



many to one cardinality

Using Sets, it can be represented as:



Set Representation of Many-to-One

Set Representation of Many-to-One

In this case, each student is taking only 1 course but 1 course has been taken by many students.

**4. Many-to-Many:** When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

the total number of tables that can be used in this is 3.

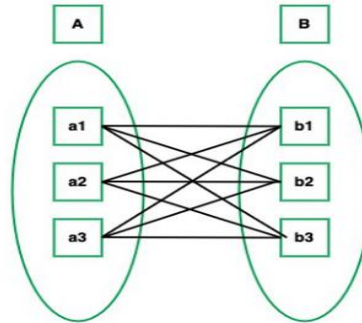


many to many cardinality



Department of CSE (Data Science)

Using Sets, it can be represented as:



Many-to-Many Set Representation

In this example, student S1 is enrolled in C1 and C3 and Course C3 is enrolled by S1, S3, and S4. So it is many-to-many relationships.

**Participation Constraint**

Participation Constraint is applied to the entity participating in the relationship set.

**1. Total Participation** – Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of students will be total. Total participation is shown by a double line in the ER diagram.

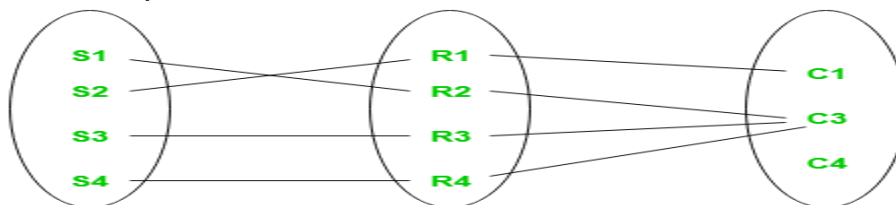
**2. Partial Participation** – The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.

The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation.



Total Participation and Partial Participation

Using Set, it can be represented as,



Set representation of Total Participation and Partial Participation



**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108

Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



---

### Department of CSE (Data Science)

---

Every student in the Student Entity set participates in a relationship but there exists a course C4 that is not taking part in the relationship.

## How to Draw ER Diagram?

- The very first step is Identifying all the Entities, and place them in a Rectangle, and labeling them accordingly.
- The next step is to identify the relationship between them and pace them accordingly using the Diamond, and make sure that, Relationships are not connected to each other.
- Attach attributes to the entities properly.
- Remove redundant entities and relationships.
- Add proper colors to highlight the data present in the database.

## Extended Entity-Relationship

EER is a high-level data model that incorporates the extensions to the original [ER model](#). Enhanced ERD are high level models that represent the requirements and complexities of complex database.

In addition to ER model concepts EE-R includes –

- Subclasses and Super classes.
- Specialization and Generalization.
- Category or union type.
- Aggregation.

These concepts are used to create EE-R diagrams.

### Subclasses and Super class

Super class is an entity that can be divided into further subtype.

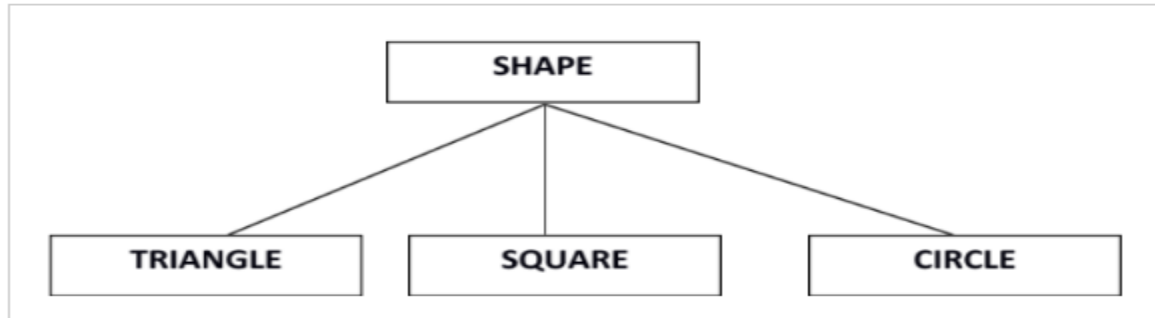
For **example** – consider Shape super class.



---

**Department of CSE (Data Science)**

---

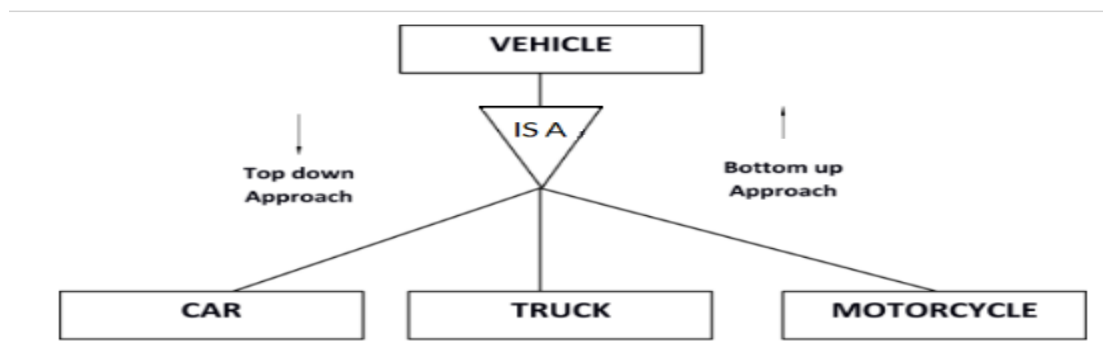


Super class shape has sub groups: Triangle, Square and Circle.

Sub classes are the group of entities with some unique attributes. Sub class inherits the properties and attributes from super class.

## Specialization and Generalization

Generalization is a process of generalizing an entity which contains generalized attributes or properties of generalized entities.



It is a Bottom up process i.e. consider we have 3 sub entities Car, Truck and Motorcycle. Now these three entities can be generalized into one super class named as Vehicle.

Specialization is a process of identifying subsets of an entity that share some different characteristic. It is a top down approach in which one entity is broken down into low level entity.



---

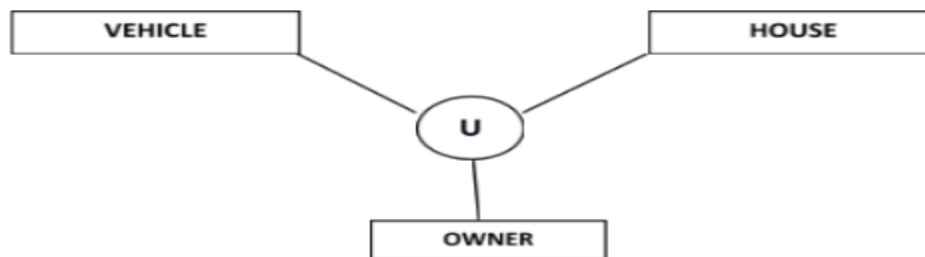
**Department of CSE (Data Science)**

---

In above example Vehicle entity can be a Car, Truck or Motorcycle.

## Category or Union

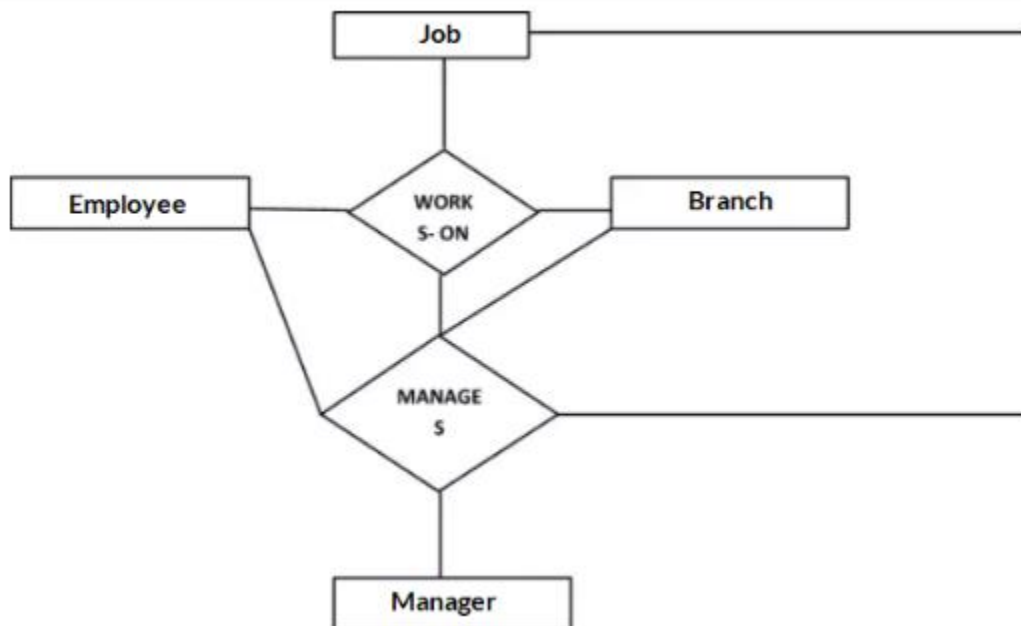
Relationship of one super or sub class with more than one super class.



Owner is the subset of two super class: Vehicle and House.

## Aggregation

Represents relationship between a whole object and its component.





---

**Department of CSE (Data Science)**

---

Consider a ternary relationship Works\_On between Employee, Branch and Manager. Now the best way to model this situation is to use aggregation, So, the relationship-set, Works\_On is a higher level entity-set. Such an entity-set is treated in the same manner as any other entity-set. We can create a binary relationship, Manager, between Works\_On and Manager to represent who manages what tasks.

## Relational Model in DBMS

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

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

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

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

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

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

**Example: STUDENT Relation**

NAME	ROLL_NO	PHONE_NO	ADDRESS	AGE
Ram	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20



---

**Department of CSE (Data Science)**

---

Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

- In the given table, NAME, ROLL\_NO, PHONE\_NO, ADDRESS, and AGE are the attributes.
- The instance of schema STUDENT has 5 tuples.
- $t_3 = \langle \text{Laxman}, 33289, 8583287182, \text{Gurugram}, 20 \rangle$

## Properties of Relations

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Attribute domain has no significance
- tuple has no duplicate value
- Order of tuple can have a different sequence

## Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

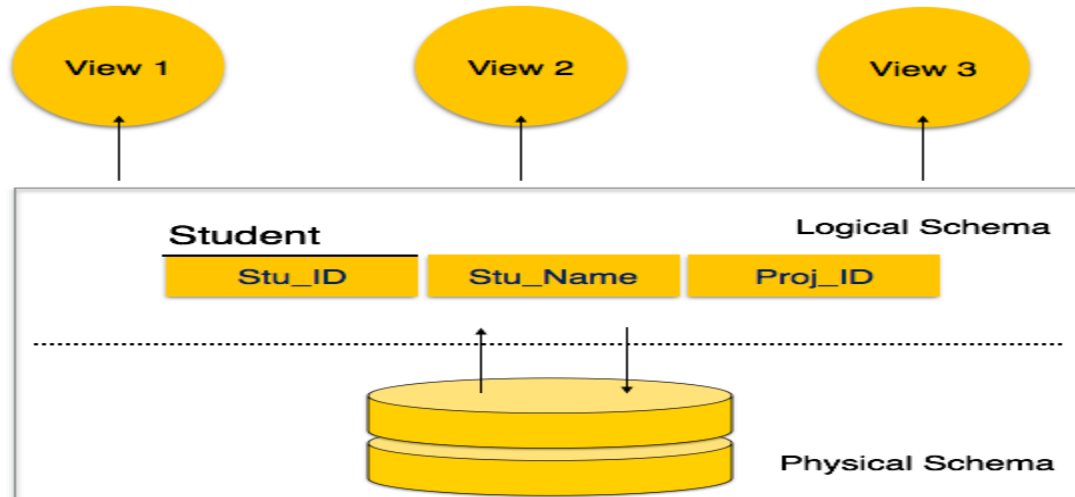
A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.



---

**Department of CSE (Data Science)**

---



A database schema can be divided broadly into two categories –

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

## Database Instance

It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

## Relational query languages

Query is a question or requesting information. Query language is a language which is used to retrieve information from a database. Relational Database





---

### Department of CSE (Data Science)

---

systems are expected to be equipped with a query language that assists users to query the database. Relational Query Language is used by the user to communicate with the database user requests for the information from the database. Relational algebra breaks the user requests and instructs the DBMS to execute the requests. It is the language by which the user communicates with the database. They are generally on a higher level than any other programming language. These relational query languages can be Procedural and Non-Procedural.

Query language is divided into two types as follows –

- Procedural language
- Non-procedural language

### Procedural language

Information is retrieved from the database by specifying the sequence of operations to be performed.

For Example: Relational algebra

Structure Query language (SQL) is based on relational algebra.

Relational algebra consists of a set of operations that take one or two relations as an input and produces a new relation as output.

The different types of relational algebra operations are –

- Select operation
- Project operation
- Rename operation
- Union operation
- Intersection operation



**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108

Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



---

### Department of CSE (Data Science)

---

- Difference operation
- Cartesian product operation
- Join operation
- Division operation.

Select, project, rename comes under unary operation (operate on one table). Union, intersection, difference, cartesian, join, division comes under binary operation (operate on two tables).

## Non-Procedural language

Information is retrieved from the database without specifying the sequence of operation to be performed. Users only specify what information is to be retrieved.

For Example: Relational Calculus

Query by Example (QBE) is based on Relational calculus

Relational calculus is a non-procedural query language in which information is retrieved from the database without specifying sequence of operation to be performed.

Relational calculus is of two types which are as follows –

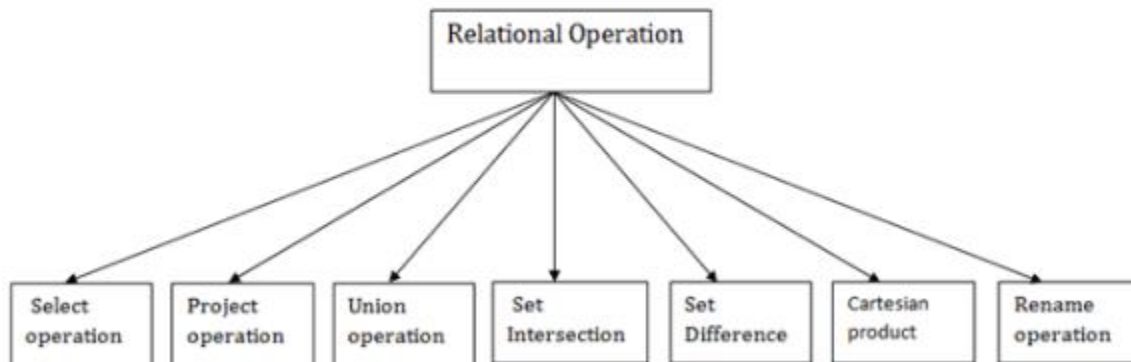
- Tuple calculus
- Domain calculus



## Relational Algebra

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

### Types of Relational operation



#### 1. Select Operation:

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

1. Notation:  $\sigma p(r)$

##### Where:

$\sigma$  is used for selection prediction

$r$  is used for relation

$p$  is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like  $=, \neq, \geq, <, >, \leq$ .

**For example: LOAN Relation**

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



**Department of CSE (Data Science)**

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

**Input:**

- $\sigma$  BRANCH\_NAME="perryride" (LOAN)

**Output:**

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

**2. Project Operation:**

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

- Notation:  $\Pi$  A1, A2, An (r)

**Where**

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

**Example: CUSTOMER RELATION**

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn



**Department of CSE (Data Science)**

Brooks	Senator	Brooklyn
--------	---------	----------

**Input:**

1.  $\square$  NAME, CITY (CUSTOMER)

**Output:**

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

**3. Union Operation:**

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

1. Notation:  $R \cup S$

A union operation must hold the following condition:

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

**Example:**

**DEPOSITOR RELATION**

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321



**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108  
Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



**Department of CSE (Data Science)**

Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

**BORROW RELATION**

<b>CUSTOMER_NAME</b>	<b>LOAN_NO</b>
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

**Input:**

1.  $\prod$  CUSTOMER\_NAME (BORROW)  $\cup$   $\prod$  CUSTOMER\_NAME (DEPOSITOR)

**Output:**

<b>CUSTOMER_NAME</b>
Johnson
Smith
Hayes
Turner
Jones



Department of CSE (Data Science)

Lindsay
Jackson
Curry
Williams
Mayes

#### 4. Set Intersection:

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

1. Notation:  $R \cap S$

**Example:** Using the above DEPOSITOR table and BORROW table

**Input:**

1.  $\cap$  CUSTOMER\_NAME (BORROW)  $\cap$   $\cap$  CUSTOMER\_NAME (DEPOSITOR)

**Output:**

CUSTOMER_NAME
Smith
Jones

#### 5. Set Difference:

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

1. Notation:  $R - S$

**Example:** Using the above DEPOSITOR table and BORROW table

**Input:**

1.  $\cap$  CUSTOMER\_NAME (BORROW) -  $\cap$  CUSTOMER\_NAME (DEPOSITOR)

**Output:**

CUSTOMER_NAME
---------------



**Department of CSE (Data Science)**

Jackson
Hayes
Willians
Curry

## 6. Cartesian product

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

1. Notation: E X D

### Example:

#### EMPLOYEE

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

#### DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

#### Input:

1. EMPLOYEE X DEPARTMENT

#### Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
--------	----------	----------	---------	-----------





**Department of CSE (Data Science)**

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

## 7. Rename Operation:

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

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

1.  $\rho(\text{STUDENT1}, \text{STUDENT})$

## Relational Calculus in DBMS

Relational calculus, a non-procedural query language in database management systems, guides users on what data is needed without specifying how to obtain it. Commonly utilized in commercial relational languages like SQL-QBE and QUEL, relational calculus ensures a focus on desired data without delving into procedural details, promoting a more efficient and abstract approach to querying in relational databases.

### What is Relational Calculus?

Before understanding Relational calculus in DBMS, we need to understand **Procedural Language** and **Declarative Language**.

1. **Procedural Language** - Those Languages which clearly define how to get the required results from the Database are called Procedural Language. **Relational algebra** is a Procedural Language.



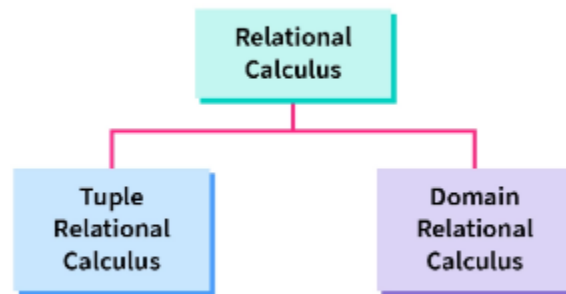
---

### Department of CSE (Data Science)

---

2. **Declarative Language** - Those Language that only cares about What to get from the database without getting into how to get the results are called Declarative Language. **Relational Calculus** is a Declarative Language.

So Relational Calculus is a Declarative Language that uses Predicate Logic or First-Order Logic to determine the results from Database.



## Types of Relational Calculus in DBMS

**Relational Calculus is of Two Types:**

1. Tuple Relational Calculus (TRC)
2. Domain Relational Calculus (DRC)

### Tuple Relational Calculus (TRC)

Tuple Relational Calculus in DBMS uses a tuple variable ( $t$ ) that goes to each row of the table and checks if the predicate is true or false for the given row. Depending on the given predicate condition, it returns the row or part of the row.

### The Tuple Relational Calculus expression Syntax

Where  $t$  is the tuple variable that runs over every Row, and  $P(t)$  is the predicate logic expression or condition.

Let's take an example of a Customer Database and try to see how TRC expressions work.



---

**Department of CSE (Data Science)**

---

*Customer Table*

Customer_id	Name	Zip code
1	Rohit	12345
2	Rahul	13245
3	Rohit	56789
4	Amit	12345.

**Example 1:** Write a TRC query to get all the data of customers whose zip code is 12345.

**TRC Query:**  $\{t \mid t \in \text{Customer} \wedge t.\text{Zipcode} = 12345\}$  or **TRC Query:**  $\{t \mid \text{Customer}(t) \wedge t[\text{Zipcode}] = 12345\}$

**Workflow of query** - The tuple variable "t" will go through every tuple of the Customer table. Each row will check whether the Cust\_Zipcode is 12345 or not and only return those rows that satisfies the Predicate expression condition.

The TRC expression above can be read as **"Return all the tuple which belongs to the Customer Table and whose Zipcode is equal to 12345."**

**Result of the TRC expression above:**

Customer_id	Name	Zip code
1	Rohit	12345
4.	Amit	12345

**Example 2:** Write a TRC query to get the customer id of all the Customers.

**TRC query:**  $\{t \mid \exists s (s \in \text{Customer} \wedge s.\text{Customer\_id} = t.\text{customer\_id})\}$

**Result of the TRC Query:**



---

**Department of CSE (Data Science)**

---

Customer_id
1
2
3
4

### Domain Relational Calculus (DRC)

Domain Relational Calculus uses domain Variables to get the column values required from the database based on the predicate expression or condition.

**The Domain relational calculus expression syntax:**

where,

$\langle x_1, x_2, x_3, x_4 \dots \rangle$  are domain variables used to get the column values required,  
and  $P(x_1, x_2, x_3 \dots)$  is predicate expression or condition.

Let's take the example of Customer Database and try to understand DRC queries with some examples.

#### Customer Table

Customer_id	Name	Zip code
1	Rohit	12345
2	Rahul	13245
3	Rohit	56789
4	Amit	12345

**Example 1:** Write a DRC query to get the data of all customers with Zip code 12345.

**DRC query:**  $\{ \langle x_1, x_2, x_3 \rangle \mid \langle x_1, x_2 \rangle \in \text{Customer} \wedge x_3 = 12345 \}$



**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108  
Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



---

**Department of CSE (Data Science)**

---

**Workflow of Query:** In the above query  $x_1, x_2, x_3$  (ordered) refers to the attribute or column which we need in the result, and the predicate condition is that the first two domain variables  $x_1$  and  $x_2$  should be present while matching the condition for each row and the third domain variable  $x_3$  should be equal to 12345.

**Result of the DRC query will be:**

Customer_id	Name	Zip code
1	Rohit	12345
4	Amit	12345

**Example 2:** Write a DRC query to get the customer id of all the customer.

**DRC Query:** {  $\langle x_1 \rangle \mid \exists x_2, x_3 (\langle x_1, x_2, x_3 \rangle \in \text{Customer})$  }

**Result of the above Query will be:**

Customer_id
1
2
3

## UNIT-II

### Functional Dependency

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

$X \rightarrow Y$

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

**For example:**



---

**Department of CSE (Data Science)**

---

Assume we have an employee table with attributes: Emp\_Id, Emp\_Name, Emp\_Address.

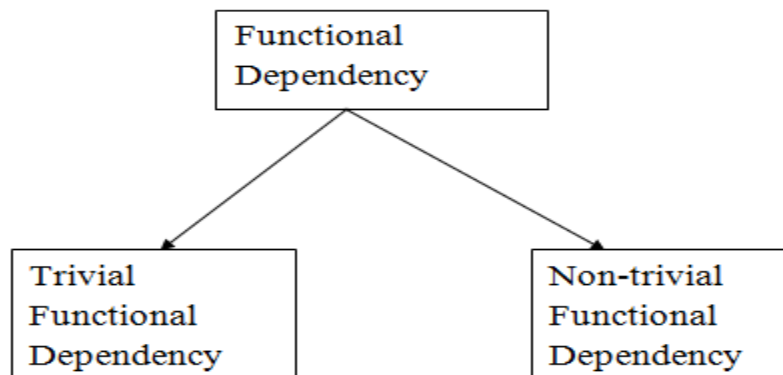
Here Emp\_Id attribute can uniquely identify the Emp\_Name attribute of employee table because if we know the Emp\_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

$\text{Emp\_Id} \rightarrow \text{Emp\_Name}$

We can say that Emp\_Name is functionally dependent on Emp\_Id.

### Types of Functional dependency



#### 1. Trivial functional dependency

- $A \rightarrow B$  has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like:  $A \rightarrow A$ ,  $B \rightarrow B$

#### Example:

Consider a table with two columns Employee\_Id and Employee\_Name.

$\{\text{Employee\_id}, \text{Employee\_Name}\} \rightarrow \text{Employee\_Id}$  is a trivial functional dependency as Employee\_Id is a subset of  $\{\text{Employee\_Id}, \text{Employee\_Name}\}$ .

Also,  $\text{Employee\_Id} \rightarrow \text{Employee\_Id}$  and  $\text{Employee\_Name} \rightarrow \text{Employee\_Name}$  are trivial dependencies too.

#### 2. Non-trivial functional dependency

- $A \rightarrow B$  has a non-trivial functional dependency if B is not a subset of A.
- When  $A \cap B$  is NULL, then  $A \rightarrow B$  is called as complete non-trivial.

#### Example:



---

**Department of CSE (Data Science)**

---

ID → Name,

Name → DOB

**Armstrong's axioms**

- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.

The Functional dependency has 6 types of inference rule:

**1. Reflexive Rule (IR<sub>1</sub>)**

In the reflexive rule, if Y is a subset of X, then X determines Y.

If  $X \supseteq Y$  then  $X \rightarrow Y$

**Example:**

$X = \{a, b, c, d, e\}$

$Y = \{a, b, c\}$

**2. Augmentation Rule (IR<sub>2</sub>)**

The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

If  $X \rightarrow Y$  then  $XZ \rightarrow YZ$

**Example:**

For R(ABCD), if  $A \rightarrow B$  then  $AC \rightarrow BC$

**3. Transitive Rule (IR<sub>3</sub>)**

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

If  $X \rightarrow Y$  and  $Y \rightarrow Z$  then  $X \rightarrow Z$

**4. Union Rule (IR<sub>4</sub>)**

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.



**Department of CSE (Data Science)**

If  $X \rightarrow Y$  and  $X \rightarrow Z$  then  $X \rightarrow YZ$

**Proof:**

$X$	$\rightarrow$	$Y$	(given)
$X$	$\rightarrow$	$Z$	(given)
$X \rightarrow XY$ (using IR <sub>2</sub> on 1 by augmentation with X. Where $XX = X$ )		$XY \rightarrow YZ$ (using IR <sub>2</sub> on 2 by augmentation with Y)	
$X \rightarrow YZ$ (using IR <sub>3</sub> on 3 and 4)			

**5. Decomposition Rule (IR<sub>5</sub>)**

Decomposition rule is also known as project rule. It is the reverse of union rule.

This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

If  $X \rightarrow YZ$  then  $X \rightarrow Y$  and  $X \rightarrow Z$

**Proof:**

$X$	$\rightarrow$	$YZ$	(given)
$YZ$	$\rightarrow$	$Y$	(using IR <sub>1</sub> Rule)
$X \rightarrow Y$ (using IR <sub>3</sub> on 1 and 2)			

**6. Pseudo transitive Rule (IR<sub>6</sub>)**

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

If  $X \rightarrow Y$  and  $YZ \rightarrow W$  then  $XZ \rightarrow W$

**Proof:**

$X$	$\rightarrow$	$Y$	(given)
$WY \rightarrow Z$ (given)			
$WX \rightarrow WY$ (using IR <sub>2</sub> on 1 by augmenting with W)			
$WX \rightarrow Z$ (using IR <sub>3</sub> on 3 and 2)			

**Normalization**

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

- Making relations very large.
- It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.





**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108  
Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



---

### Department of CSE (Data Science)

---

- The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

#### What is Normalization?

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

#### Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

#### Data modification anomalies can be categorized into three types:

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

#### Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

#### Following are the various types of Normal forms:



**Department of CSE (Data Science)**

Normal Form	Description
<u>1NF</u>	A relation is in 1NF if it contains an atomic value.
<u>2NF</u>	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
<u>3NF</u>	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.
<u>4NF</u>	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
<u>5NF</u>	A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

**Advantages of Normalization**

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

**Disadvantages of Normalization**

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.



---

**Department of CSE (Data Science)**

---

**First Normal Form (1NF)**

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

**Example:** Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP\_PHONE.

**EMPLOYEE table:**

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar



**Department of CSE (Data Science)**

12	Sam	7390372389	Punjab
12	Sam	8589830302	Punja

**Second Normal Form (2NF)**

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

**Example:** Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

**TEACHER table**

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER\_AGE is dependent on TEACHER\_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

**TEACHER\_DETAIL table:**



**Department of CSE (Data Science)**

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

**TEACHER\_SUBJECT table:**

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

**Third Normal Form (3NF)**

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency  $X \rightarrow Y$ .



---

**Department of CSE (Data Science)**

---

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

**Example:**

**EMPLOYEE\_DETAIL table:**

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

**Super key in the table above:**

1. {EMP\_ID}, {EMP\_ID, EMP\_NAME}, {EMP\_ID, EMP\_NAME, EMP\_ZIP}....so on

**Candidate key:** {EMP\_ID}

**Non-prime attributes:** In the given table, all attributes except EMP\_ID are non-prime.

Here, EMP\_STATE & EMP\_CITY dependent on EMP\_ZIP and EMP\_ZIP dependent on EMP\_ID. The non-prime attributes (EMP\_STATE, EMP\_CITY) transitively dependent on super key(EMP\_ID). It violates the rule of third normal form.

That's why we need to move the EMP\_CITY and EMP\_STATE to the new <EMPLOYEE\_ZIP> table, with EMP\_ZIP as a Primary key.

**EMPLOYEE table:**



**Department of CSE (Data Science)**

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

**EMPLOYEE\_ZIP table:**

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

**Boyce Codd normal form (BCNF)**

- BCNF is the advance version of 3NF. It is stricter than 3NF.



---

**Department of CSE (Data Science)**

---

- A table is in BCNF if every functional dependency  $X \rightarrow Y$ ,  $X$  is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

**Example:** Let's assume there is a company where employees work in more than one department.

**EMPLOYEE table:**

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

**In the above table Functional dependencies are as follows:**

1.  $EMP\_ID \rightarrow EMP\_COUNTRY$
2.  $EMP\_DEPT \rightarrow \{DEPT\_TYPE, EMP\_DEPT\_NO\}$

**Candidate key: {EMP-ID, EMP-DEPT}**

The table is not in BCNF because neither  $EMP\_DEPT$  nor  $EMP\_ID$  alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

**EMP\_COUNTRY table:**

EMP_ID	EMP_COUNTRY
--------	-------------





**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108

Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



**Department of CSE (Data Science)**

264	India
264	India

**EMP\_DEPT table:**

<b>EMP_DEPT</b>	<b>DEPT_TYPE</b>	<b>EMP_DEPT_NO</b>
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

**EMP\_DEPT\_MAPPING table:**

<b>EMP_ID</b>	<b>EMP_DEPT</b>
D394	283
D394	300
D283	232
D283	549



---

**Department of CSE (Data Science)**

---

**Functional dependencies:**

1. EMP\_ID → EMP\_COUNTRY
2. EMP\_DEPT → {DEPT\_TYPE, EMP\_DEPT\_NO}

**Candidate keys:**

For the first table: EMP\_ID  
For the second table: EMP\_DEPT  
For the third table: {EMP\_ID, EMP\_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

**Fourth normal form (4NF)**

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

**Example**

**STUDENT**

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey



---

**Department of CSE (Data Science)**

---

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU\_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU\_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

**STUDENT\_COURSE**

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

**STUDENT\_HOBBY**

STU_ID	HOBBY
21	Dancing
21	Singing



**Department of CSE (Data Science)**

34	Dancing
74	Cricket
59	Hockey

**Fifth normal form (5NF)**

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

**Example**

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.



---

**Department of CSE (Data Science)**

---

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

**P1**

<b>SEMESTER</b>	<b>SUBJECT</b>
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

**P2**

<b>SUBJECT</b>	<b>LECTURER</b>
Computer	Anshika
Computer	John
Math	John
Math	Akash



**Department of CSE (Data Science)**

Chemistry	Praveen
-----------	---------

**P3**

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

**Multivalued Dependency**

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

**Example:** Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
------------	------------	-------



**Department of CSE (Data Science)**

M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF\_YEAR are dependent on BIKE\_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE\_MODEL. The representation of these dependencies is shown below:

1. BIKE\_MODEL  $\twoheadrightarrow$  MANUF\_YEAR
2. BIKE\_MODEL  $\twoheadrightarrow$  COLOR

This can be read as "BIKE\_MODEL multidetermined MANUF\_YEAR" and "BIKE\_MODEL multidetermined COLOR".

### Closure of an Attribute

**Closure of an Attribute:** Closure of an Attribute can be defined as a set of attributes that can be functionally determined from it.

OR

Closure of a set F of FDs is the set F<sup>+</sup> of all FDs that can be inferred from F



---

**Department of CSE (Data Science)**

---

Closure of a set of attributes X concerning F is the set  $X^+$  of all attributes that are functionally determined by X

Pseudocode to find Closure of an Attribute?

Determine  $X^+$ , the closure of X under functional dependency set F

X Closure := will contain X itself;

Repeat the process as:

old X Closure := X Closure;

for each functional dependency  $P \rightarrow Q$  in FD set do

if X Closure is subset of P then X Closure := X Closure  $\cup$  Q ;

Repeat until ( X Closure = old X Closure);

Algorithm of Determining  $X^+$ , the Closure of X under F

**Input:** A set F of FDs on a relation schema R, and a set of attributes X, which is a subset of R.

$X^+ := X$ ;

repeat

old $X^+ := X^+$  ;

for each functional dependency  $Y \rightarrow Z$  in F do

if  $X^+ \supseteq Y$  then  $X^+ := X^+ \cup Z$ ;

until ( $X^+ = \text{old}X^+$ );

### Lossless Decomposition

Lossless join decomposition is a decomposition of a relation R into relations R1, and R2 such that if we perform a natural join of relation R1 and R2, it will return the original relation R. This is effective in removing redundancy from databases while preserving the original data.

In other words by lossless decomposition, it becomes feasible to reconstruct the relation R from decomposed tables R1 and R2 by using Joins.

Only **1NF, 2NF, 3NF**, and **BCNF** are valid for lossless join decomposition.





---

### Department of CSE (Data Science)

---

In Lossless Decomposition, we select the common attribute and the criteria for selecting a common attribute is that the common attribute must be a candidate key or super key in either relation R1, R2, or both.

Decomposition of a relation R into R1 and R2 is a lossless-join decomposition if at least one of the following functional dependencies is in F+ (Closure of functional dependencies)

#### Example of Lossless Decomposition

— Employee (Employee\_Id, Ename, Salary, Department\_Id, Dname)

Can be decomposed using lossless decomposition as,

— Employee\_desc (Employee\_Id, Ename, Salary, Department\_Id)

— Department\_desc (Department\_Id, Dname)

Alternatively the lossy decomposition would be as joining these tables is not possible so not possible to get back original data.

– Employee\_desc (Employee\_Id, Ename, Salary)

– Department\_desc (Department\_Id, Dname)

R1  $\cap$  R2  $\rightarrow$  R1

OR

R1  $\cap$  R2  $\rightarrow$  R2

In a [database management system \(DBMS\)](#), a lossless decomposition is a process of decomposing a relation schema into multiple relations in such a way that it preserves the information contained in the original relation. Specifically, a lossless decomposition is one in which the original relation can be reconstructed by joining the decomposed relations.

To achieve lossless decomposition, a set of conditions known as Armstrong's axioms can be used. These conditions ensure that the decomposed relations will retain all the information present in the original relation. Specifically, the two most important axioms for lossless decomposition are the reflexivity and the decomposition axiom.

The reflexivity axiom states that if a set of attributes is a subset of another set of attributes, then the larger set of attributes can be inferred from the smaller set. The decomposition axiom states that if a relation R can be decomposed into two relations R1 and R2, then the original relation R can be reconstructed by taking the natural join of R1 and R2.

There are several algorithms available for performing lossless decomposition in DBMS, such as the [BCNF \(Boyce-Codd Normal Form\)](#) decomposition and the [3NF \(Third Normal Form\)](#) decomposition. These algorithms use a set of rules to decompose a relation into multiple relations while ensuring that the original relation can be reconstructed without any loss of information.

#### Advantages of Lossless Decomposition

1. **Reduced Data Redundancy:** Lossless decomposition helps in reducing the data redundancy that exists in the original relation. This helps in improving the efficiency of the database system by reducing storage requirements and improving query performance.
2. **Maintenance and Updates:** Lossless decomposition makes it easier to maintain and update the database since it allows for more granular control over the data.



---

### Department of CSE (Data Science)

---

3. **Improved Data Integrity:** Decomposing a relation into smaller relations can help to improve data integrity by ensuring that each relation contains only data that is relevant to that relation. This can help to reduce data inconsistencies and errors.
4. **Improved Flexibility:** Lossless decomposition can improve the flexibility of the database system by allowing for easier modification of the schema.

#### Disadvantages of Lossless Decomposition

- **Increased Complexity:** Lossless decomposition can increase the complexity of the database system, making it harder to understand and manage.
- **Increased Processing Overhead:** The process of decomposing a relation into smaller relations can result in increased processing overhead. This can lead to slower query performance and reduced efficiency.
- **Join Operations:** Lossless decomposition may require additional join operations to retrieve data from the decomposed relations. This can also result in slower query performance.
- **Costly:** Decomposing relations can be costly, especially if the database is large and complex. This can require additional resources, such as hardware and personnel.

#### Dependency Preserving Decomposition

If we decompose a relation R into relations R1 and R2, All dependencies of R either must be a part of R1 or R2 or must be derivable from a combination of [functional dependency](#) of R1 and R2. For Example, A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of R1(ABC).

#### Advantages of Lossless Join and Dependency Preserving Decomposition

- **Improved Data Integrity:** Lossless join and dependency preserving decomposition help to maintain the data integrity of the original relation by ensuring that all dependencies are preserved.
- **Reduced Data Redundancy:** These techniques help to reduce [data redundancy](#) by breaking down a relation into smaller, more manageable relations.
- **Improved Query Performance:** By breaking down a relation into smaller, more focused relations, query performance can be improved.
- **Easier Maintenance and Updates:** The smaller, more focused relations are easier to maintain and update than the original relation, making it easier to modify the database schema and update the data.
- **Better Flexibility:** Lossless join and dependency preserving decomposition can improve the flexibility of the database system by allowing for easier modification of the schema.

#### Disadvantages of Lossless Join and Dependency Preserving Decomposition

- **Increased Complexity:** Lossless join and dependency-preserving decomposition can increase the complexity of the database system, making it harder to understand and manage.
- **Costly:** Decomposing relations can be costly, especially if the database is large and complex. This can require additional resources, such as hardware and personnel.



**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108  
Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



---

### Department of CSE (Data Science)

---

- **Reduced Performance:** Although query performance can be improved in some cases, in others, lossless join and dependency-preserving decomposition can result in reduced query performance due to the need for additional join operations.
- **Limited Scalability:** These techniques may not scale well in larger databases, as the number of smaller, focused relations can become unwieldy.

## Unit 3

### Query Processing

Query Processing is the activity performed in extracting data from the database. In query processing, it takes various steps for fetching the data from the database. The steps involved are:

1. Parsing and translation
2. Optimization
3. Evaluation

The query processing works in the following way:

#### Parsing and Translation

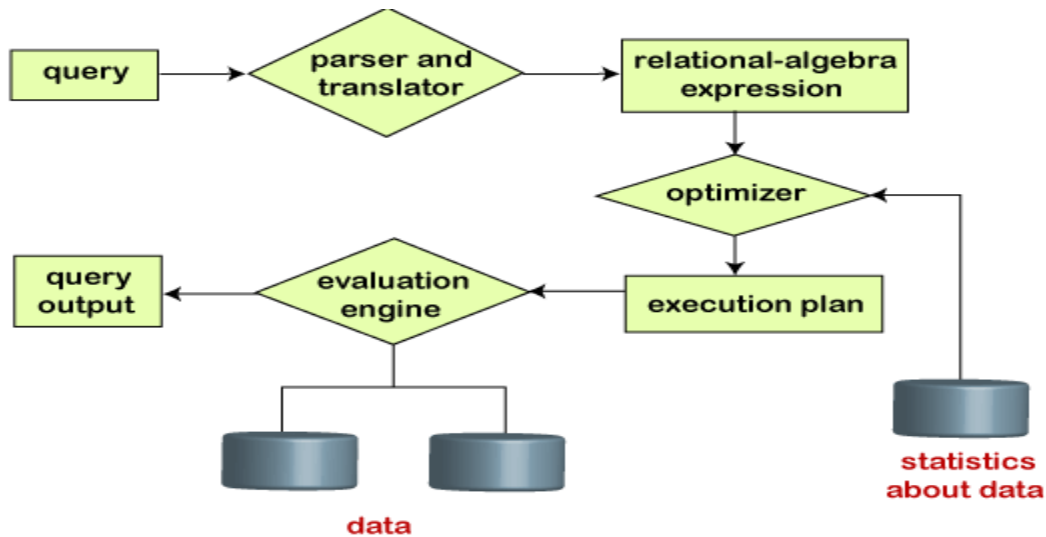
As query processing includes certain activities for data retrieval. Initially, the given user queries get translated in high-level database languages such as SQL. It gets translated into expressions that can be further used at the physical level of the file system. After this, the actual evaluation of the queries and a variety of query -optimizing transformations and takes place. Thus before processing a query, a computer system needs to translate the query into a human-readable and understandable language. Consequently, SQL or Structured Query Language is the best suitable choice for humans. But, it is not perfectly suitable for the internal representation of the query to the system. Relational algebra is well suited for the internal representation of a query. The translation process in query processing is similar to the parser of a query. When a user executes any query, for generating the internal form of the query, the parser in the system checks the syntax of the query, verifies the name of the relation in the database, the tuple, and finally the required attribute value.



Department of CSE (Data Science)

The parser creates a tree of the query, known as 'parse-tree.' Further, translate it into the form of relational algebra. With this, it evenly replaces all the use of the views when used in the query.

Thus, we can understand the working of a query processing in the below-described diagram:



Steps in query processing

Suppose a user executes a query. As we have learned that there are various methods of extracting the data from the database. In SQL, a user wants to fetch the records of the employees whose salary is greater than or equal to 10000. For doing this, the following query is undertaken:

**select emp\_name from Employee where salary>10000;**

Thus, to make the system understand the user query, it needs to be translated in the form of relational algebra. We can bring this query in the relational algebra form as:

- $\sigma_{\text{salary}>10000} (\pi_{\text{salary}} (\text{Employee}))$
- $\pi_{\text{salary}} (\sigma_{\text{salary}>10000} (\text{Employee}))$

After translating the given query, we can execute each relational algebra operation by using different algorithms. So, in this way, a query processing begins its working.

Evaluation



**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108

Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



---

### Department of CSE (Data Science)

---

For this, with addition to the relational algebra translation, it is required to annotate the translated relational algebra expression with the instructions used for specifying and evaluating each operation. Thus, after translating the user query, the system executes a query evaluation plan.

#### Query Evaluation Plan

- In order to fully evaluate a query, the system needs to construct a query evaluation plan.
- The annotations in the evaluation plan may refer to the algorithms to be used for the particular index or the specific operations.
- Such relational algebra with annotations is referred to as **Evaluation Primitives**. The evaluation primitives carry the instructions needed for the evaluation of the operation.
- Thus, a query evaluation plan defines a sequence of primitive operations used for evaluating a query. The query evaluation plan is also referred to as **the query execution plan**.
- A **query execution engine** is responsible for generating the output of the given query. It takes the query execution plan, executes it, and finally makes the output for the user query.

#### Optimization

- The cost of the query evaluation can vary for different types of queries. Although the system is responsible for constructing the evaluation plan, the user does need not to write their query efficiently.
- Usually, a database system generates an efficient query evaluation plan, which minimizes its cost. This type of task performed by the database system and is known as Query Optimization.
- For optimizing a query, the query optimizer should have an estimated cost analysis of each operation. It is because the overall operation cost depends on the memory allocations to several operations, execution costs, and so on.



**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108

Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



---

## Department of CSE (Data Science)

---

Finally, after selecting an evaluation plan, the system evaluates the query and produces the output of the query.

### Evaluation of relational algebra Expressions

We are already aware of computing and representing the individual relational operations for the given user query or expression. Here, we will get to know how to compute and evaluate an expression with multiple operations.

For evaluating an expression that carries multiple operations in it, we can perform the computation of each operation one by one. However, in the query processing system, we use two methods for evaluating an expression carrying multiple operations. These methods are:

1. Materialization
2. Pipelining

Let's take a brief discussion of these methods.

#### Materialization

In this method, the given expression evaluates one relational operation at a time. Also, each operation is evaluated in an appropriate sequence or order. After evaluating all the operations, the outputs are materialized in a temporary relation for their subsequent uses. It leads the materialization method to a disadvantage. The disadvantage is that it needs to construct those temporary relations for materializing the results of the evaluated operations, respectively. These temporary relations are written on the disks unless they are small in size.

#### Pipelining

Pipelining is an alternate method or approach to the materialization method. In pipelining, it enables us to evaluate each relational operation of the expression simultaneously in a pipeline. In this approach, after evaluating one operation, its output is passed on to the next operation, and the chain continues till all the relational operations are evaluated thoroughly. Thus, there is no requirement of storing a temporary relation in pipelining. Such an advantage of pipelining makes it a better approach as compared to the approach used in the materialization method. Even the costs of both approaches can have subsequent differences in-between. But, both approaches perform the best role in different cases. Thus, both ways are feasible at their place.



**TULSIRAMJI GAIKWAD-PATIL College of Engineering and Technology**

Wardha Road, Nagpur-441108

Accredited with NAAC A+ Grade

Approved by AICTE, New Delhi, Govt. of Maharashtra

(An Autonomous Institution Affiliated to RTM Nagpur University)



---

### Department of CSE (Data Science)

---

We have described and discussed the materialization as well as pipelining method deeply in our next sections one by one.

#### Selection Operation in Query Processing

Generally, the selection operation is performed by the file scan. **File scans** are the search algorithms that are used for locating and accessing the data. It is the lowest-level operator used in query processing.

In RDBMS or relational database systems, the file scan reads a relation only if the whole relation is stored in one file only. When the selection operation is performed on a relation whose tuples are stored in one file, it uses the following algorithms:

- **Linear Search:** In a linear search, the system scans each record to test whether satisfying the given selection condition. For accessing the first block of a file, it needs an initial seek. If the blocks in the file are not stored in contiguous order, then it needs some extra seeks. However, linear search is the slowest algorithm used for searching, but it is applicable in all types of cases. This algorithm does not care about the nature of selection, availability of indices, or the file sequence. But other algorithms are not applicable in all types of cases.
- **Index-based search:** In that algorithms are known as **Index scans**. Such index structures are known as **access paths**. These paths allow locating and accessing the data in the file. There are following algorithms that use the index in query processing:
  - **Primary index, equality on a key:** We use the index to retrieve a single record that satisfies the equality condition for making the selection. The equality comparison is performed on the key attribute carrying a primary key.



---

**Department of CSE (Data Science)**

---

- **Primary index, equality on nonkey:** The difference between equality on key and nonkey is that in this, we can fetch multiple records. We can fetch multiple records through a primary key when the selection criteria specify the equality comparison on a nonkey.
- **Secondary index, equality on key or nonkey:** The selection that specifies an equality condition can use the secondary index. Using secondary index strategy, we can either retrieve a single record when equality is on key or multiple records when the equality condition is on nonkey. When retrieving a single record, the time cost is equal to the primary index. In the case of multiple records, they may reside on different blocks. This results in one I/O operation per fetched record, and each I/O operation requires a seek and a block transfer.

### Join algorithms in Database

There are two algorithms to compute natural join and conditional join of two relations in database: Nested loop join, and Block nested loop join.

To understand these algorithms we will assume there are two relations, relation R and relation S. Relation R has  $T_R$  tuples and occupies  $B_R$  blocks. Relation S has  $T_S$  tuples and occupies  $B_S$  blocks. We will also assume relation R is the outer relation and S is the inner relation.

#### Nested Loop Join

In the nested loop join algorithm, for each tuple in outer relation, we have to compare it with all the tuples in the inner relation then only the next tuple of outer relation is considered. All pairs of tuples which satisfy the condition are added in the result of the join.

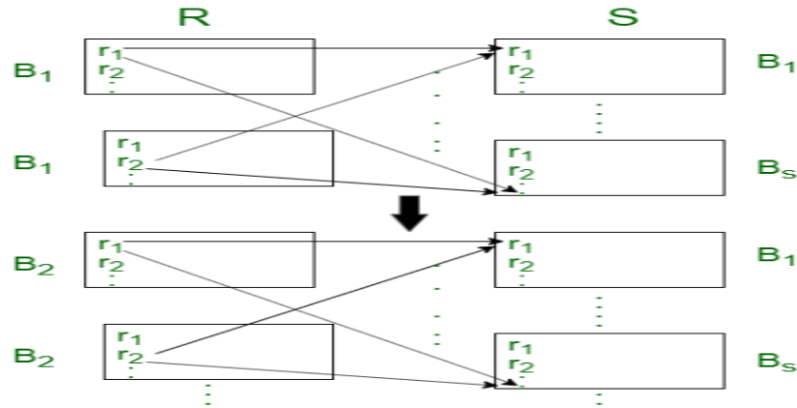
```
for each tuple  $t_R$  in  $T_R$  do  
  for each tuple  $t_s$  in  $T_s$  do  
    compare ( $t_R$ ,  $t_s$ ) if they satisfies the condition  
    add them in the result of the join  
  end  
end
```

This algorithm is called nested join because it consists of nested for loops.





Department of CSE (Data Science)



Let's see some cases to understand the performance of this algorithm,

**Case-1:** Assume only two blocks of main memory are available to store blocks from R and S relation.

For each tuple in relation to R, we have to transfer all blocks of relation S and each block of relation R should be transferred only once.  
So, the total block transfers needed =  $T_R * B_S + B_R$

**Block Nested Loop Join:**

In block nested loop join, for a block of outer relation, all the tuples in that block are compared with all the tuples of the inner relation, then only the next block of outer relation is considered. All pairs of tuples which satisfy the condition are added in the result of the join.

```

for each block  $b_R$  in  $B_R$  do
  for each block  $b_S$  in  $B_S$  do
    for each tuple  $t_R$  in  $T_R$  do
      for each tuple  $t_S$  in  $T_S$  do
        compare  $(t_R, t_S)$  if they satisfies the condition
        add them in the result of the join
      end
    end
  end
end

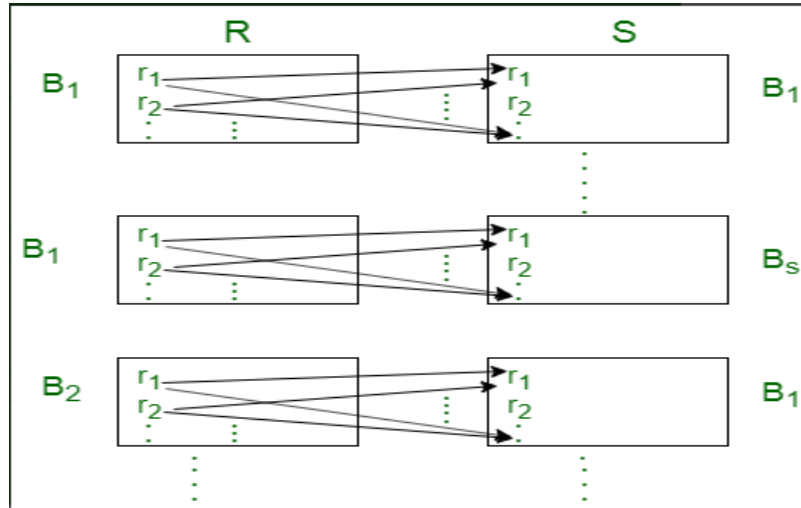
```

Let's look at some similar cases as nested loop join,

**Case-1:** Assume only two blocks of main memory are available to store blocks from R and S relation.



Department of CSE (Data Science)



For each block of relation R, we have to transfer all blocks of relation S and each block of relation R should be transferred only once. So, the total block transfers needed =  $B_R + B_R * B_S$

### Heuristic optimization in DBMS

Cost-based optimization is expensive. Heuristics are used to reduce the number of choices that must be made in a cost-based approach.

#### Rules

Heuristic optimization transforms the expression-tree by using a set of rules which improve the performance. These rules are as follows –

- Perform the SELECTION process foremost in the query. This should be the first action for any SQL table. By doing so, we can decrease the number of records required in the query, rather than using all the tables during the query.
- Perform all the projection as soon as achievable in the query. Somewhat like a selection but this method helps in decreasing the number of columns in the query.
- Perform the most restrictive joins and selection operations. What this means is that select only those sets of tables and/or views which will result in a relatively lesser number of



---

### Department of CSE (Data Science)

---

records and are extremely necessary in the query. Obviously any query will execute better when tables with few records are joined.

Some systems use only heuristics and the others combine heuristics with partial cost-based optimization.

Steps in heuristic optimization

Let's see the steps involve in heuristic optimization, which are explained below –

- Deconstruct the conjunctive selections into a sequence of single selection operations.
- Move the selection operations down the query tree for the earliest possible execution.
- First execute those selections and join operations which will produce smallest relations.
- Replace the cartesian product operation followed by selection operation with join operation.
- Deconstructive and move the tree down as far as possible.
- Identify those subtrees whose operations are pipelined.

### Materialized View

Materialized views are also known as virtual tables, but the result of the query expression is saved in physical memory. The query definition is also stored in the database. We can also consider them a Physical copy of the original base tables. It is primarily used in the context of warehousing of data. There is no standard view to define materialized view in SQL. However, few database management systems offer custom extensions to use materialized views. Unlike the normal view, they are not updated each time they are used. Instead, we need to update it manually or with the help of the trigger. The process of updating the Materialized view is known as Materialized View Maintenance.

It stores the result in the physical memory, it responds faster than the normal view because the normal view is created whenever we run the query. It is mainly used for summarizing, pre-computing, replicating and distributing data, etc.

Let's understand the syntax of the materialized view.

1. Create Materialized View view\_name



---

**Department of CSE (Data Science)**

---

2. Build [clause] Refresh [ type]
3. ON [trigger ]
4. As <query expression>

In the above syntax, the Build clause decides when to populate the materialized view. It contains two options -

- **IMMEDIATE** - It populate the materialized view immediately.
- **DEFERRED** - Need to refresh materialized view manually at least once.

Refresh type define the how to update the materialized view. There are three options -

- **FAST** - The materialized view logs is required against the source table in advance, without logs, the creation fails. A fast refresh is attempted. A fast refresh is attempted.
- **COMPLETE** - The table segment supporting the materialized view is truncated and repopulated completely using the associated query.
- **FORCE** - The materialized logs is not required. A fast refresh is attempted.

On trigger defines when to update the materialized view. The refresh can be triggered in the two ways -

- **ON COMMIT** - When the data change is committed in one of the dependent tables. The refresh is triggered.
- **ON DEMAND** - A refresh happens when we schedule task or a manual request.