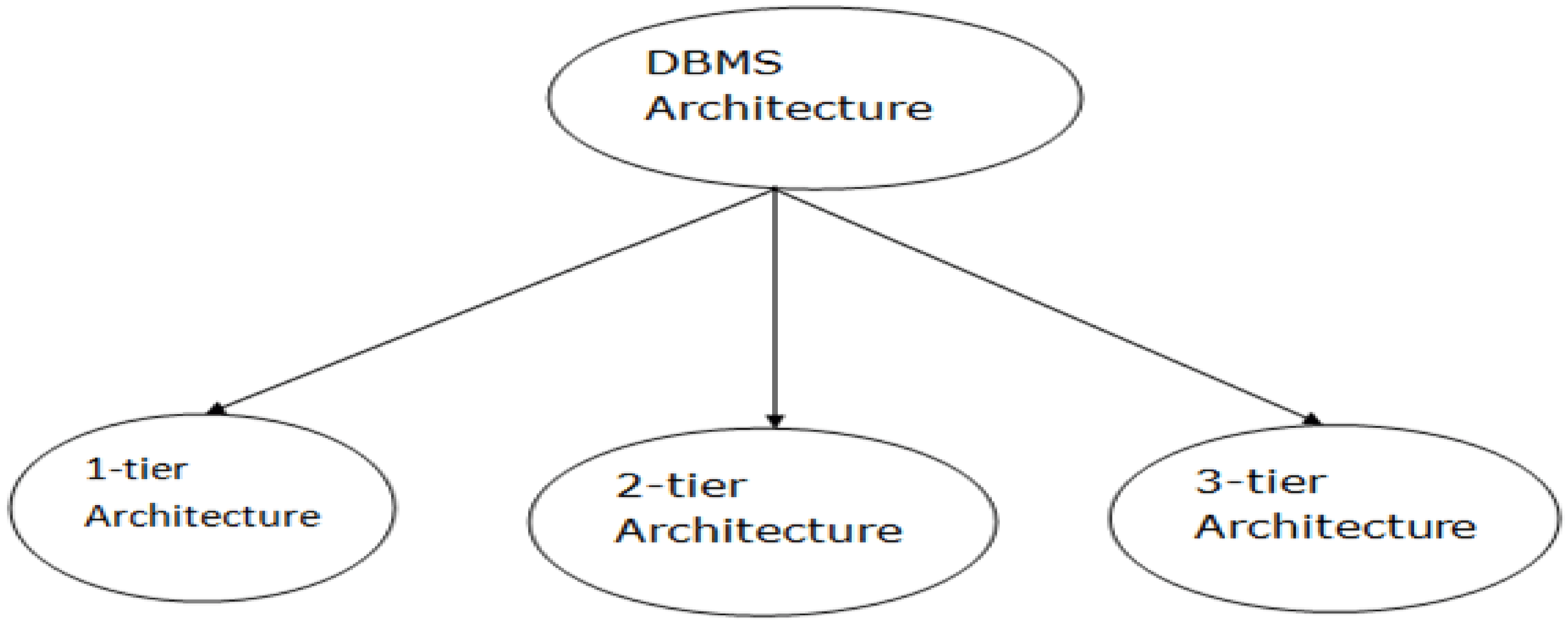
DBMS vs. File System

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

DBMS Architecture

- The DBMS design depends upon its architecture.
- The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network.
- DBMS architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Architecture



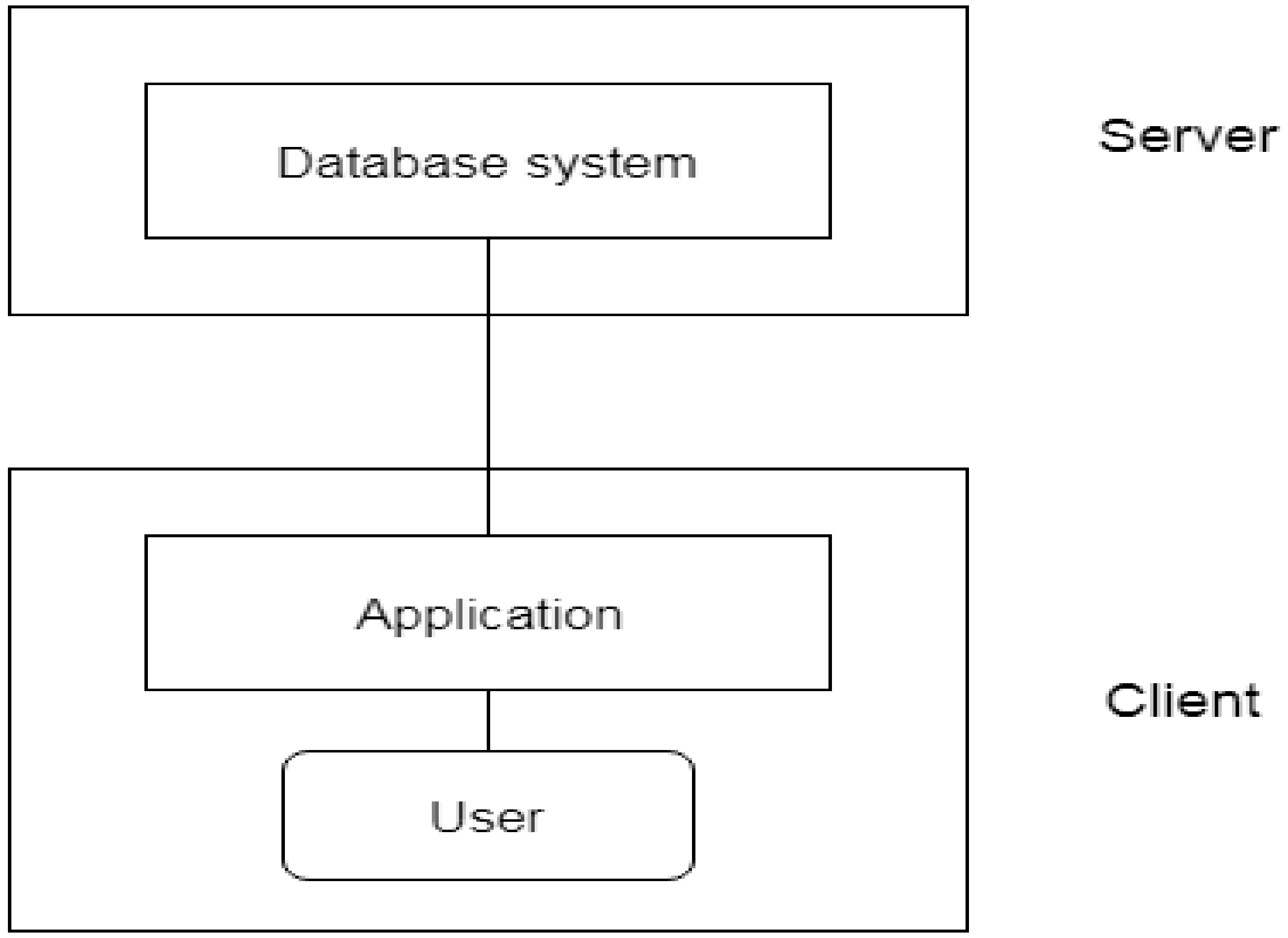
1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

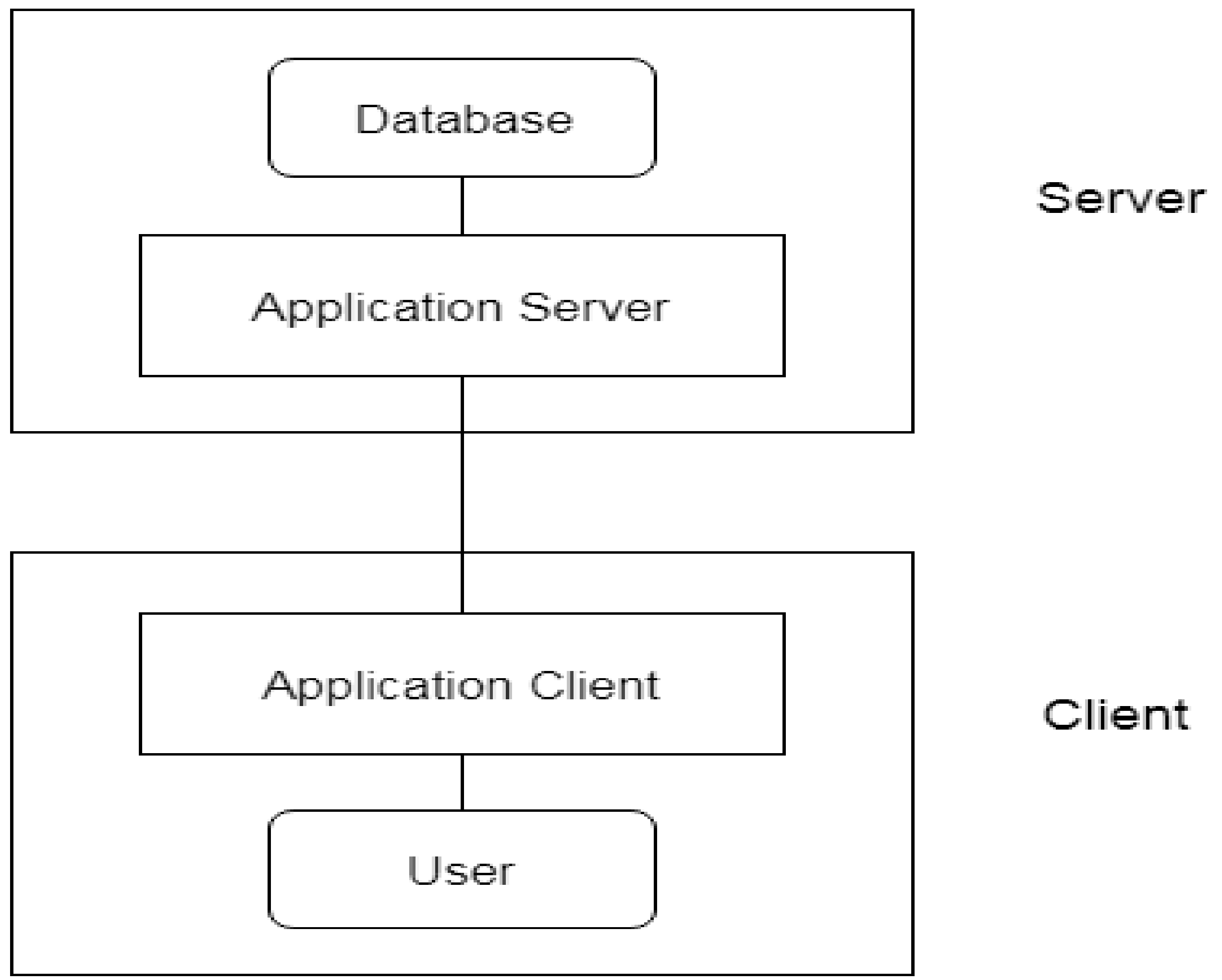
Fig: 2-tier Architecture



3-Tier Architecture

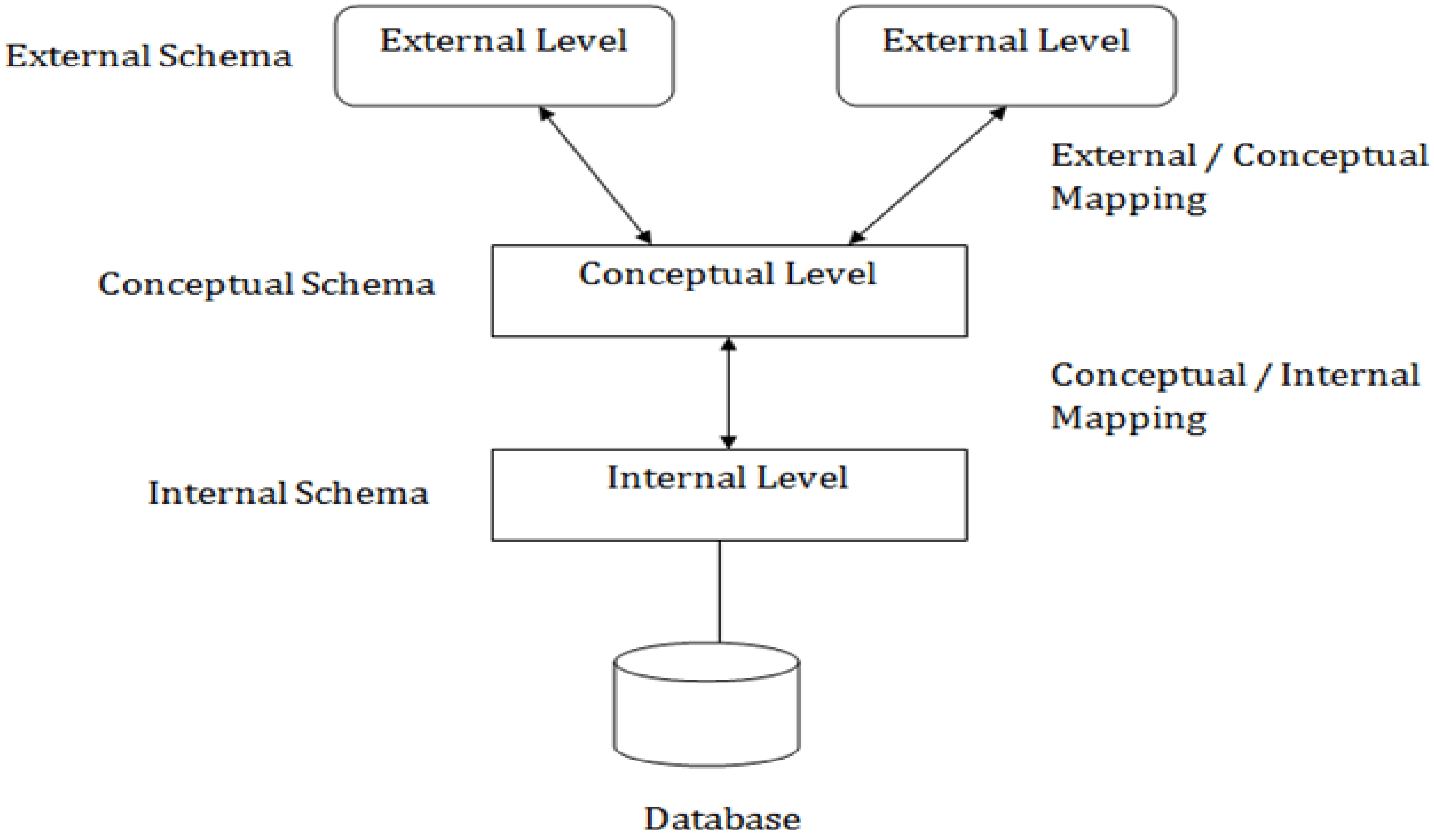
- The 3-Tier architecture contains another layer between the client and server.
- In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server.
- The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

Fig: 3-tier Architecture



Three schema Architecture

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


External Schema
External Level
External Level
External / Conceptual Mapping
Conceptual Schema
Conceptual Level
Conceptual / Internal Mapping
Internal Schema
Internal Level
Database

In the above diagram:

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

1. Internal Level

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

2. Conceptual Level

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

3. External Level

- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- An external schema is also known as view schema.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems.

Unit-1, DBMS

- Database management system is software that is used to manage the database.
- Our topics of DBMS such as introduction, ER model, keys, relational model, join operation, SQL, functional dependency, transaction, concurrency control, etc.

what is data?

- Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.
- Word 'Data' is originated from the word 'datum' that means 'single piece of information.' It is plural of the word datum.
- In computing, Data is information that can be translated into a form for efficient movement and processing. Data is interchangeable.

What is Database?

- A **database** is an organized collection of data, so that it can be easily accessed and managed.
- You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.
- **Database handlers** create a database in such a way that only one set of software program provides access of data to all the users.
- The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data.
- There are many **dynamic websites** on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.
- There are many **databases available** like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.
- Modern databases are managed by the database management system (DBMS).

Evolution of Databases

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

The Evolution

- File-Based
- 1968 was the year when File-Based database were introduced. In file-based databases, data was maintained in a flat file. Though files have many advantages, there are several limitations.
- One of the major advantages is that the file system has various access methods, e.g., sequential, indexed, and random.
- It requires extensive programming in a third-generation language such as COBOL, BASIC.
- Hierarchical Data Model
- 1968-1980 was the era of the Hierarchical Database. Prominent hierarchical database model was IBM's first DBMS. It was called IMS (Information Management System).
- In this model, files are related in a parent/child manner.
- Below diagram represents Hierarchical Data Model. Small circle represents objects.

• Database Management System

- Database management system is a software which is used to manage the database. For example: MySQL, Oracle, etc are a very popular commercial database which is used in different applications.
- DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
- It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

DBMS allows users the following tasks:

- **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
- **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.
- **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
- **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

Characteristics of DBMS

- It uses a digital repository established on a server to store and manage the information.
- It can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contains ACID properties which maintain data in a healthy state in case of failure.
- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirements of the user.

Advantages of DBMS

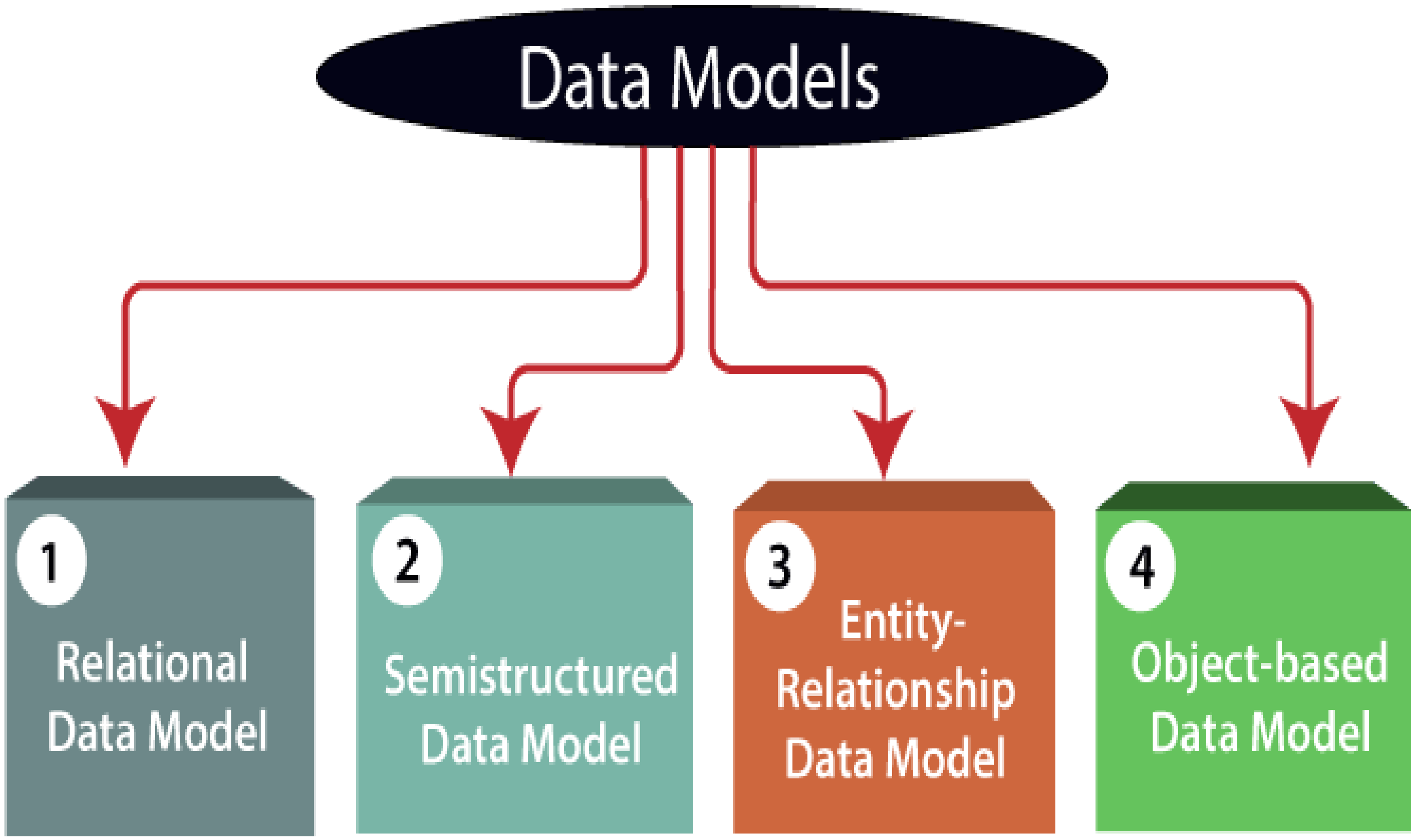
- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
- **Size:** It occupies a large space of disks and large memory to run them efficiently.
- **Complexity:** Database system creates additional complexity and requirements.
- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

Data Models

- Data Model is the modeling of the data description, data semantics, and consistency constraints of the data.
- It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following four data models used for understanding the structure of the database:


Data Models
1
Relational Data Model
2
Semistructured Data Model
3
Entity-Relationship Data Model
4
Object-based Data Model

1. Relational Data Model

- This type of model designs the data in the form of rows and columns within a table.
- Thus, a relational model uses tables for representing data and in-between relationships.
- Tables are also called relations.
- This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

2. Entity-Relationship Data Model

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

3 Object-based Data Model

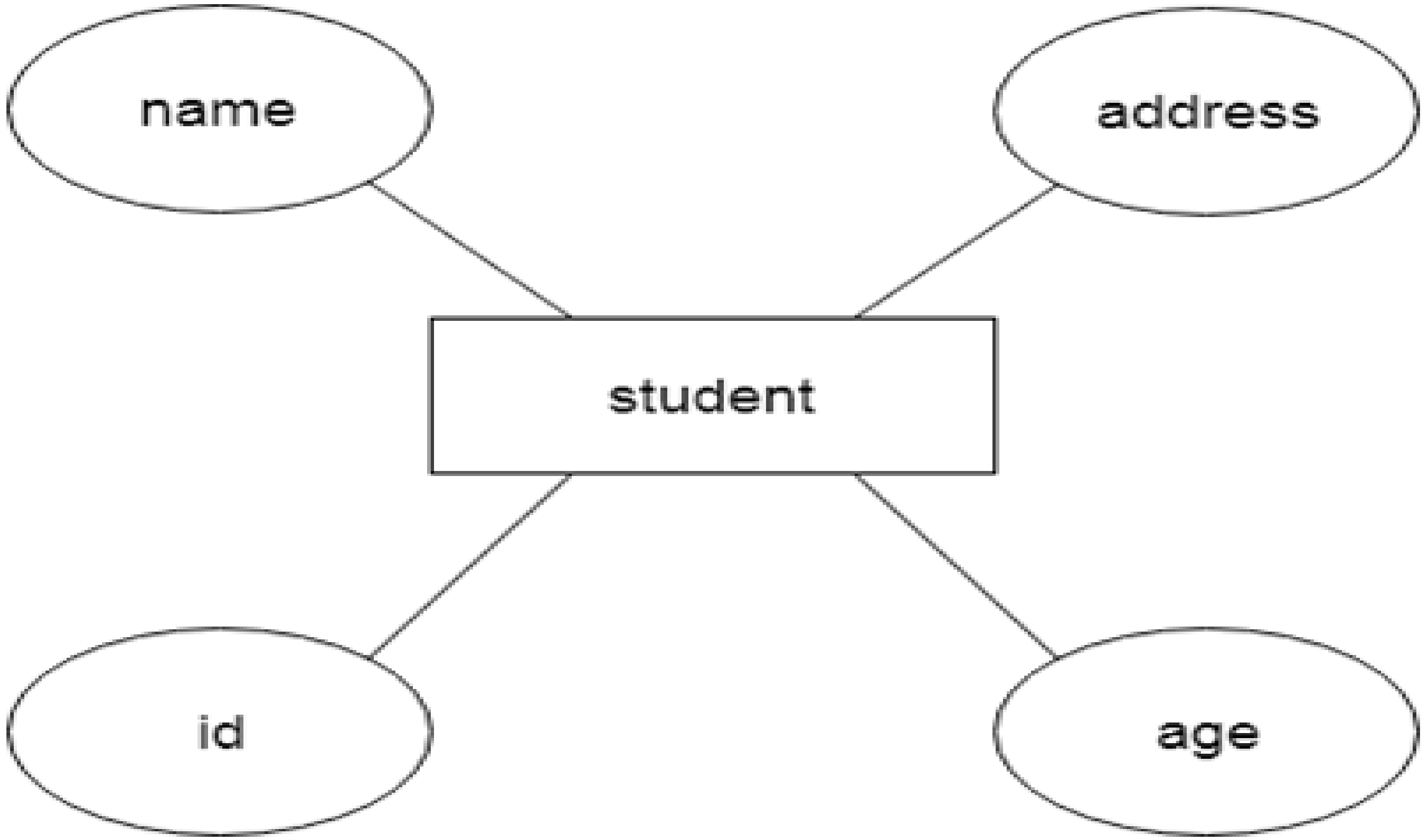
- An extension of the ER model with notions of functions, encapsulation, and object identity, as well.
- This model supports a rich type system that includes structured and collection types.
- Thus, in 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.

4. Semistructured Data Model

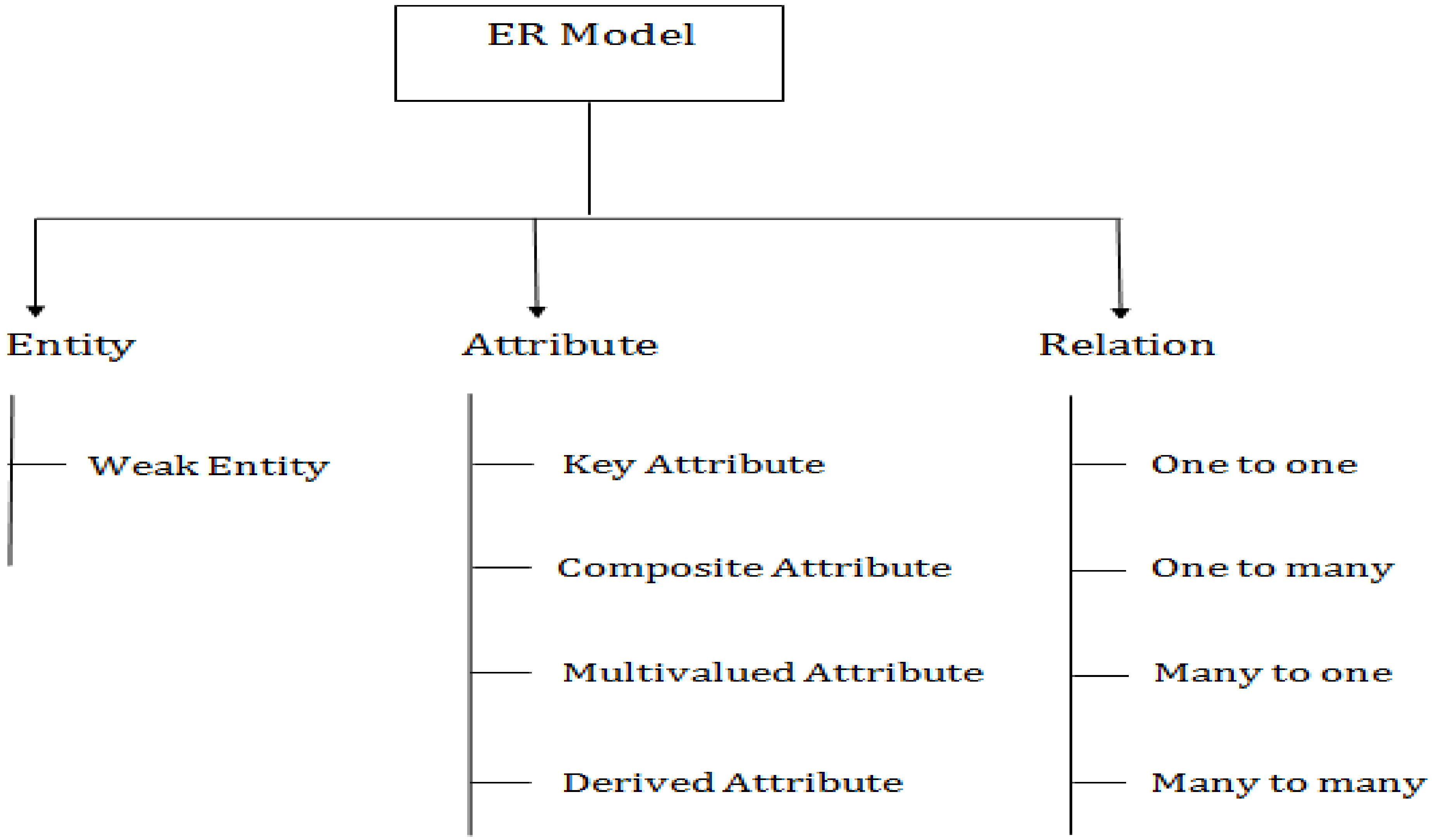
- This type of data model is different from the other three data models (explained above).
- The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets.
- The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data. Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

ER model

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.
- **For example,** Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.

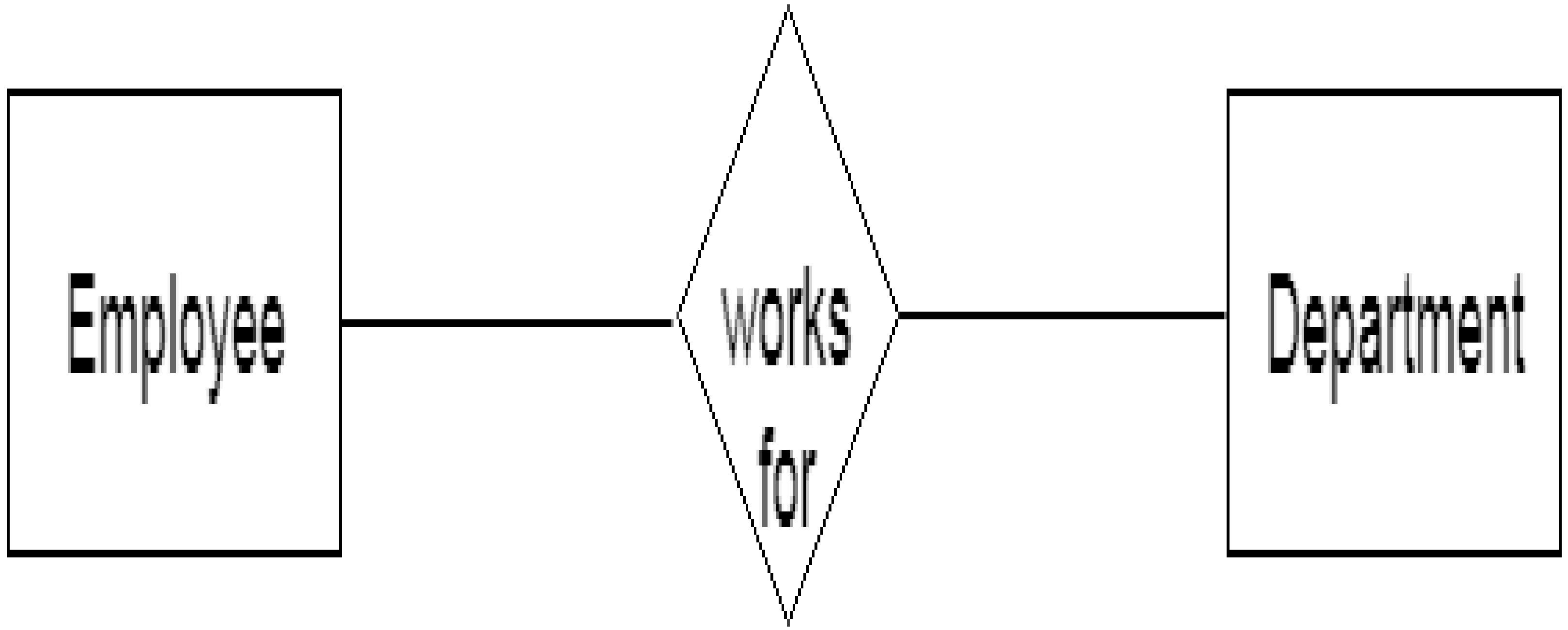

name
address
student
id
age

Component of ER Diagram



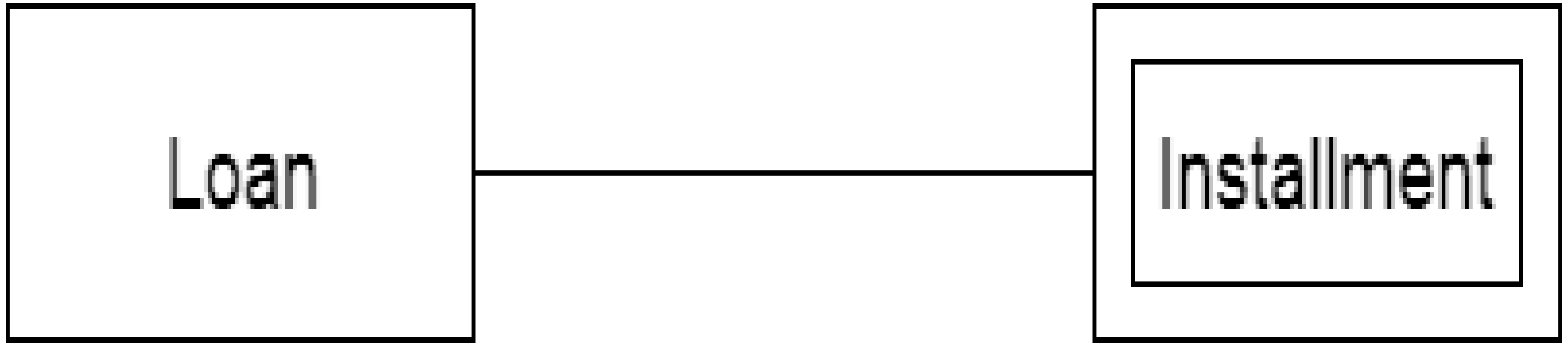
1. Entity:

- An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

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

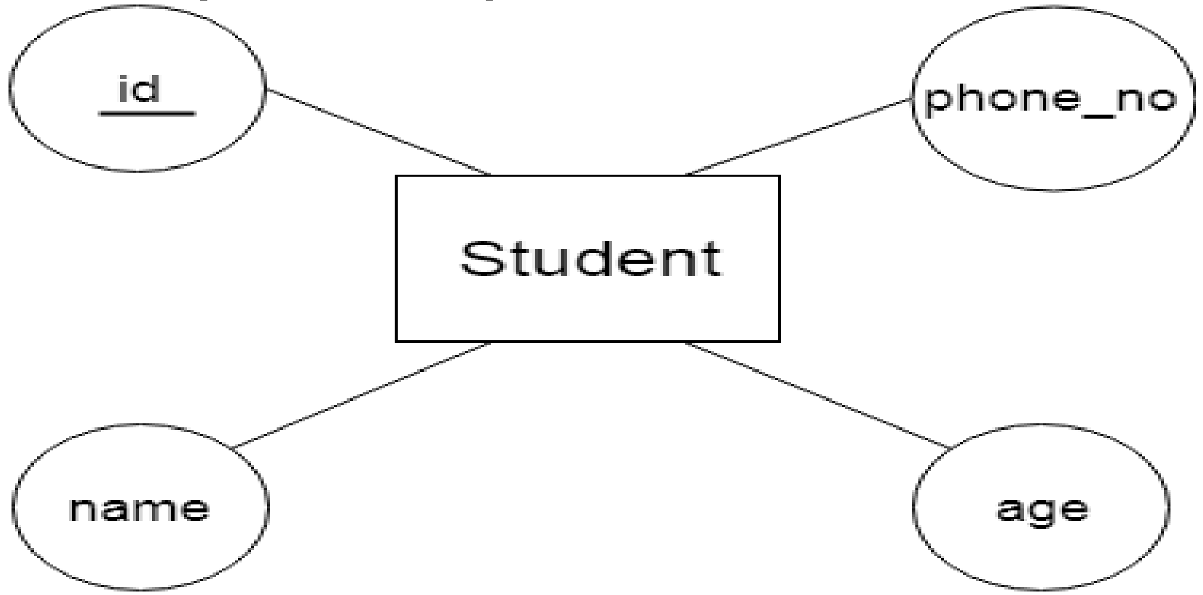


2. Attribute

- The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
- **For example,** id, age, contact number, name, etc. can be attributes of a student.

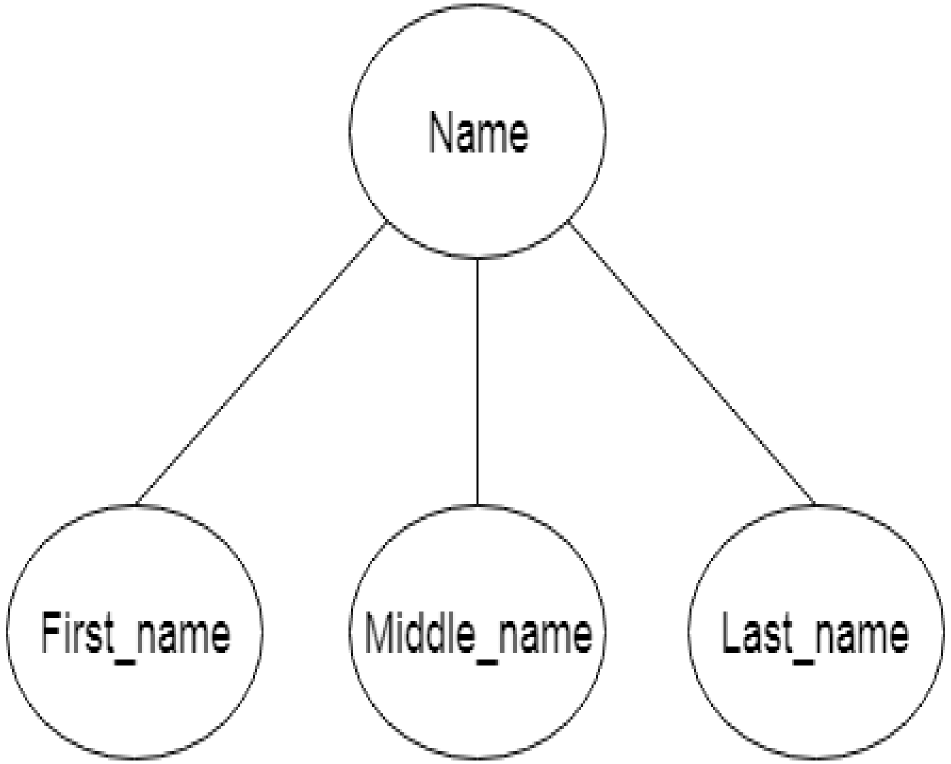
a. Key Attribute

- The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



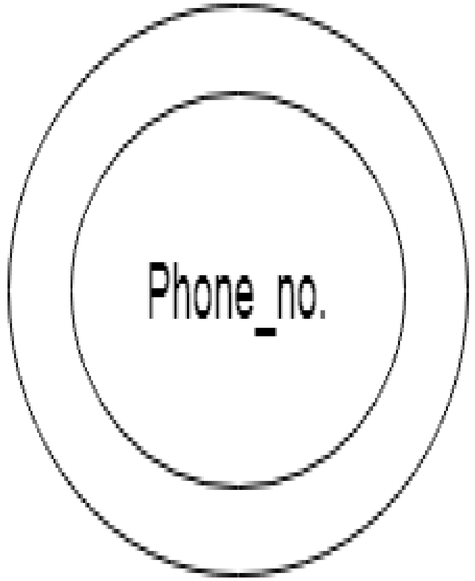
b. Composite Attribute

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



c. Multivalued Attribute

- An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.
- **For example,** a student can have more than one phone number.
-

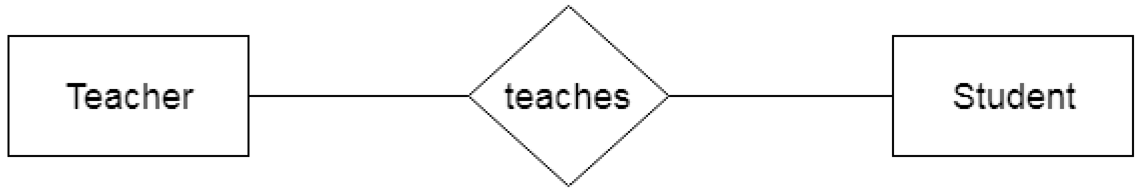


d. Derived Attribute

- An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.
- **For example,** A person's age changes over time and can be derived from another attribute like Date of birth.

3. Relationship

- A relationship is used to describe the relation between entities. Diamond is used to represent the relationship.

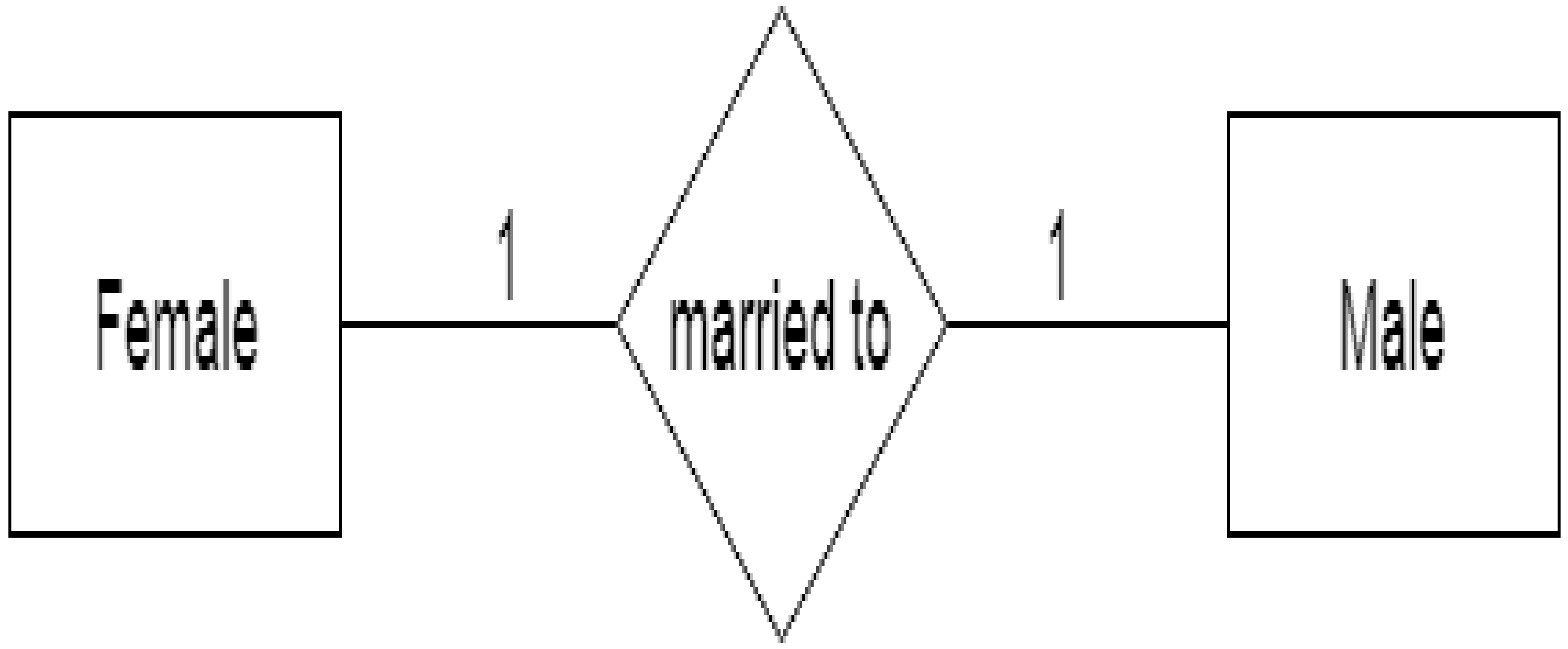


Types of relationship are as follows:

- **a. One-to-One Relationship**
- **b. One-to-many relationship**
- **c. Many-to-one relationship**
- **d. Many-to-many relationship**

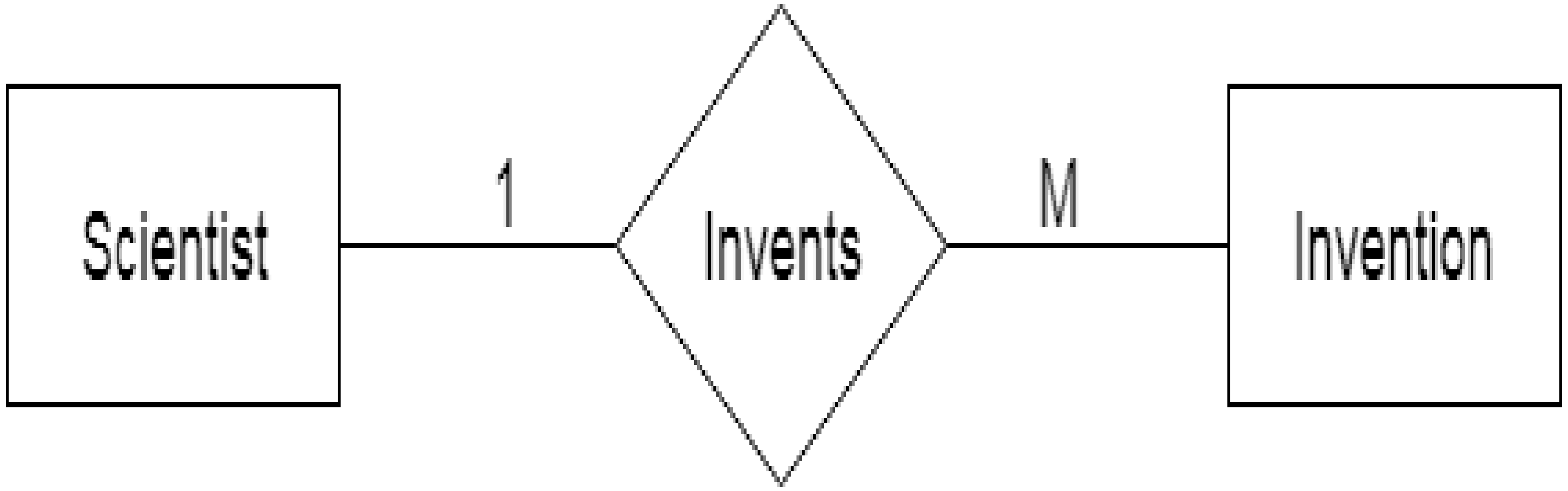
a. One-to-One Relationship

- When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.
- **For example,** A female can marry to one male, and a male can marry to one female.



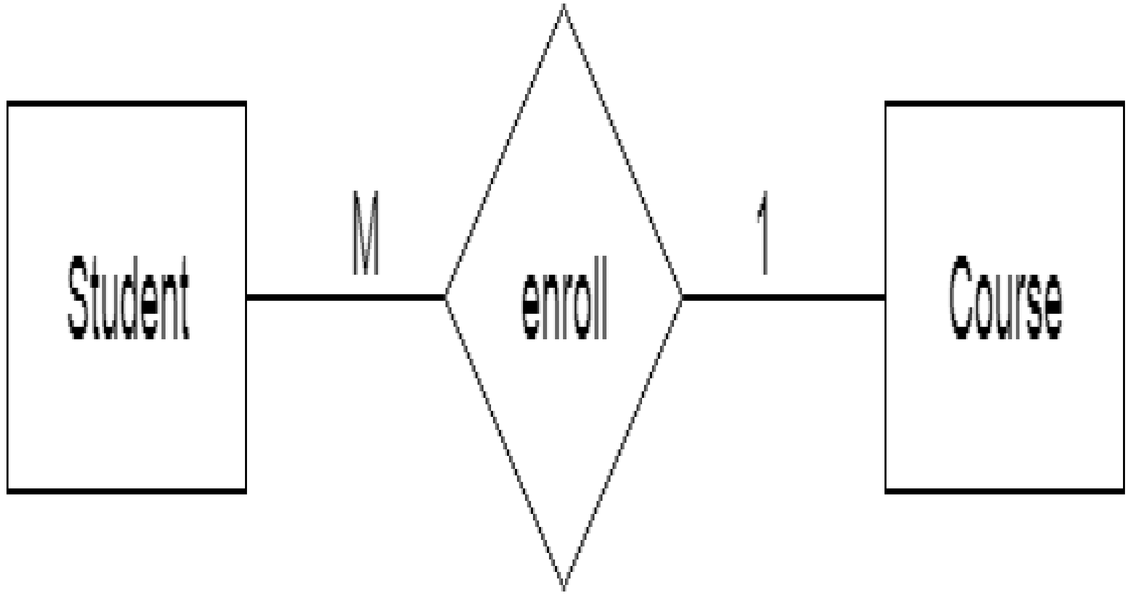
b. One-to-many relationship

- When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one -to-many relationship.
- **For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



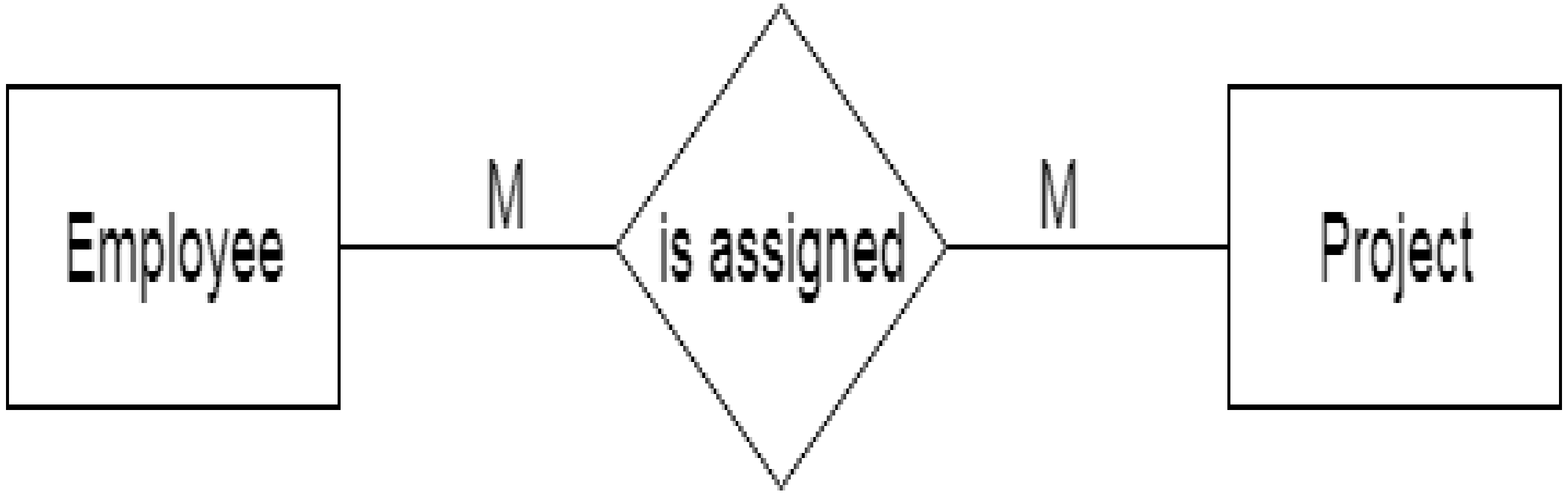
c. Many-to-one relationship

- When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.
- **For example,** Student enrolls for only one course, but a course can have many students.
-

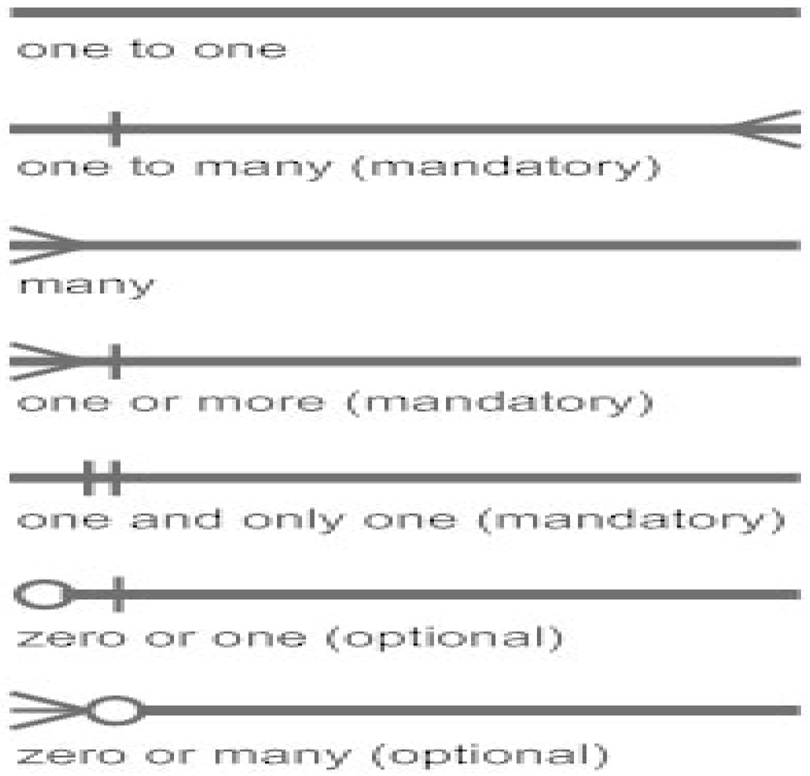


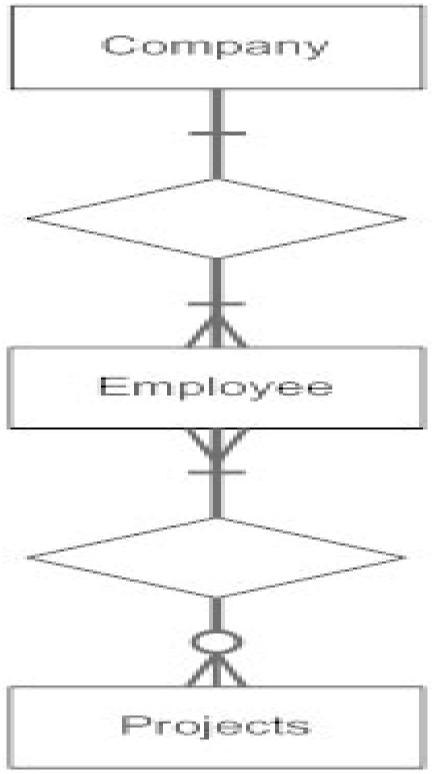
d. Many-to-many relationship

- When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.
- **For example,** Employee can assign by many projects and project can have many employees.



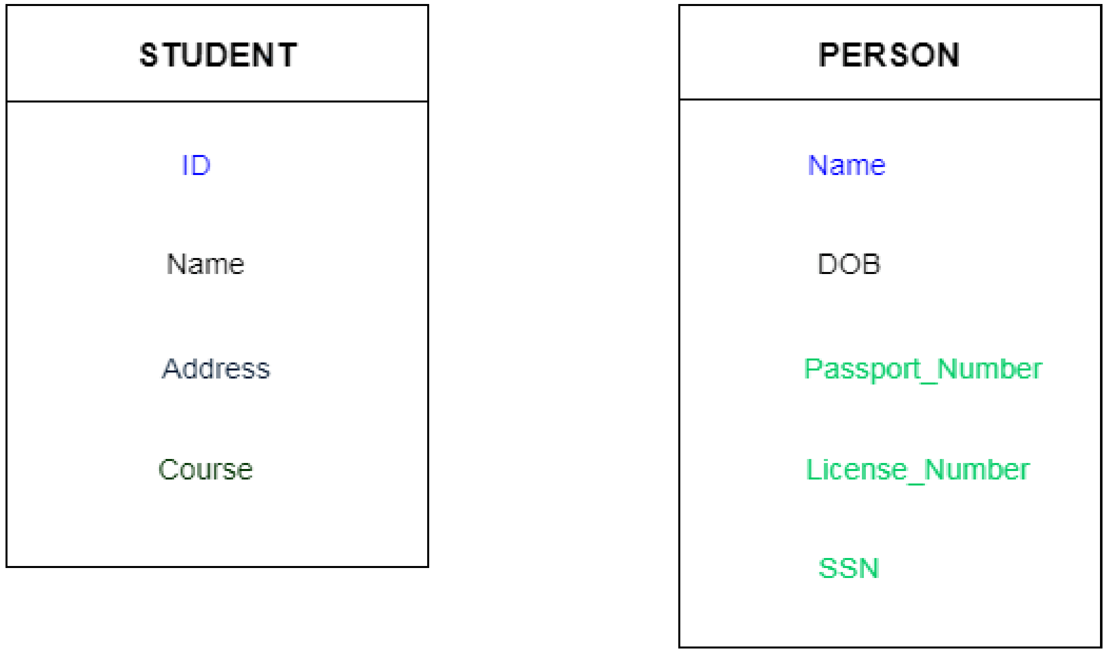
Notation of ER diagram



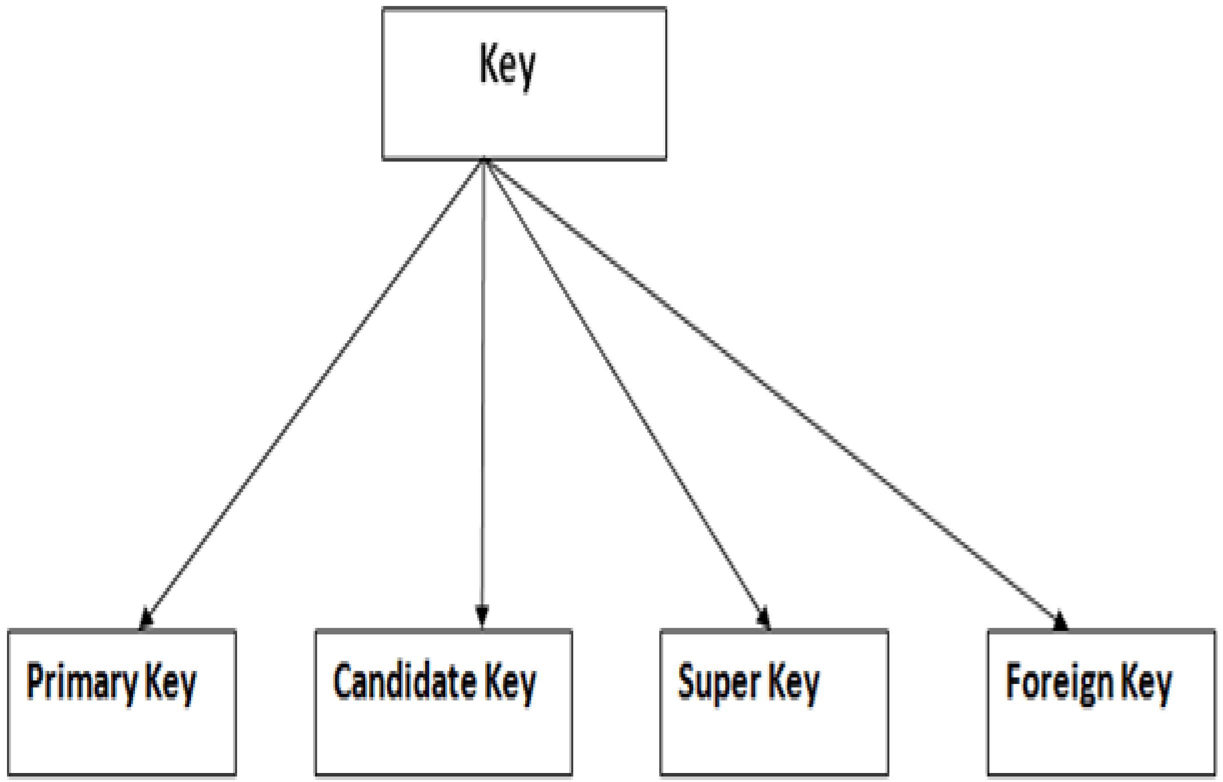


Keys

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.
- **For example:** In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

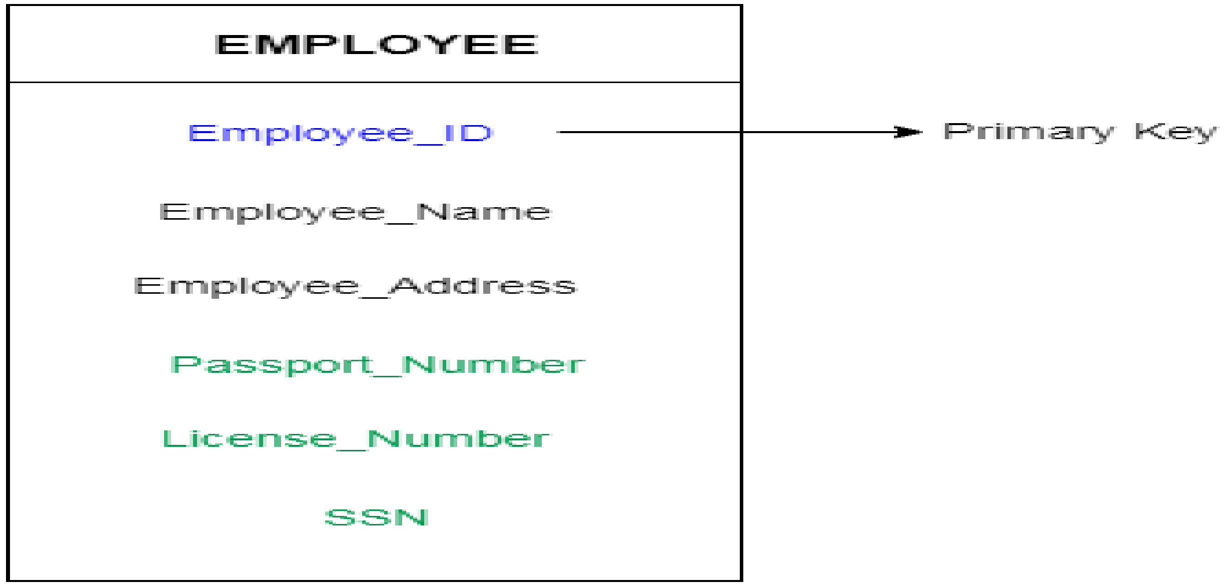


Types of key:



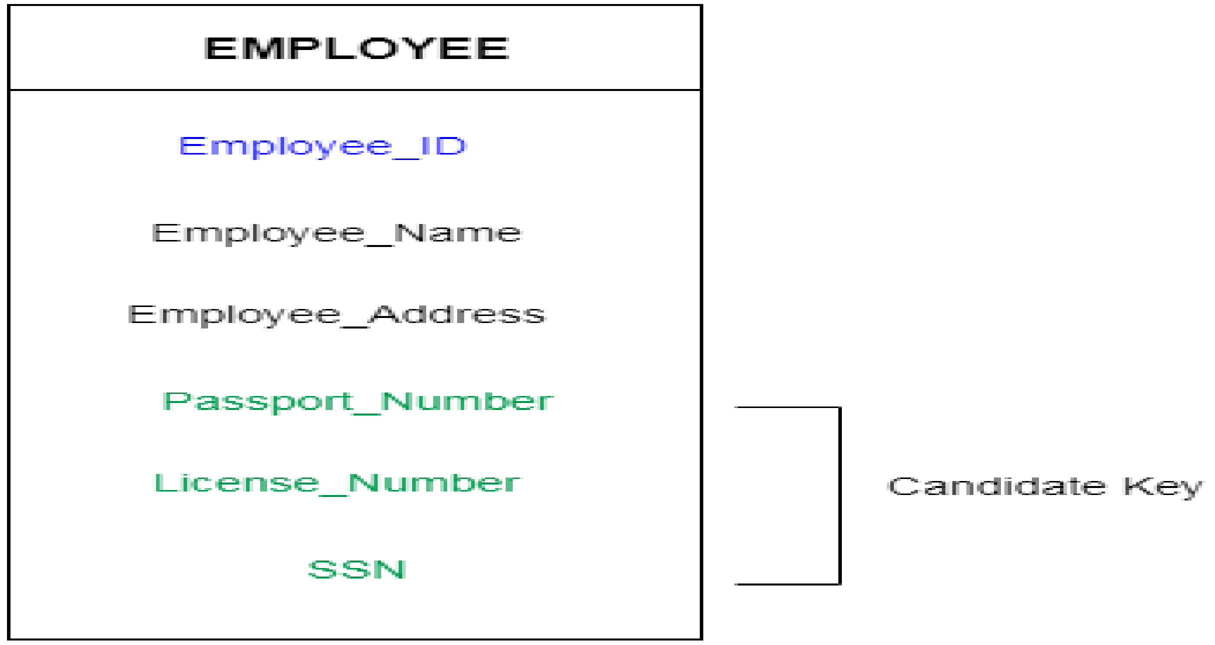
1. Primary key

- It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.
- In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary key since they are also unique.
- For each entity, selection of the primary key is based on requirement and developers**.**



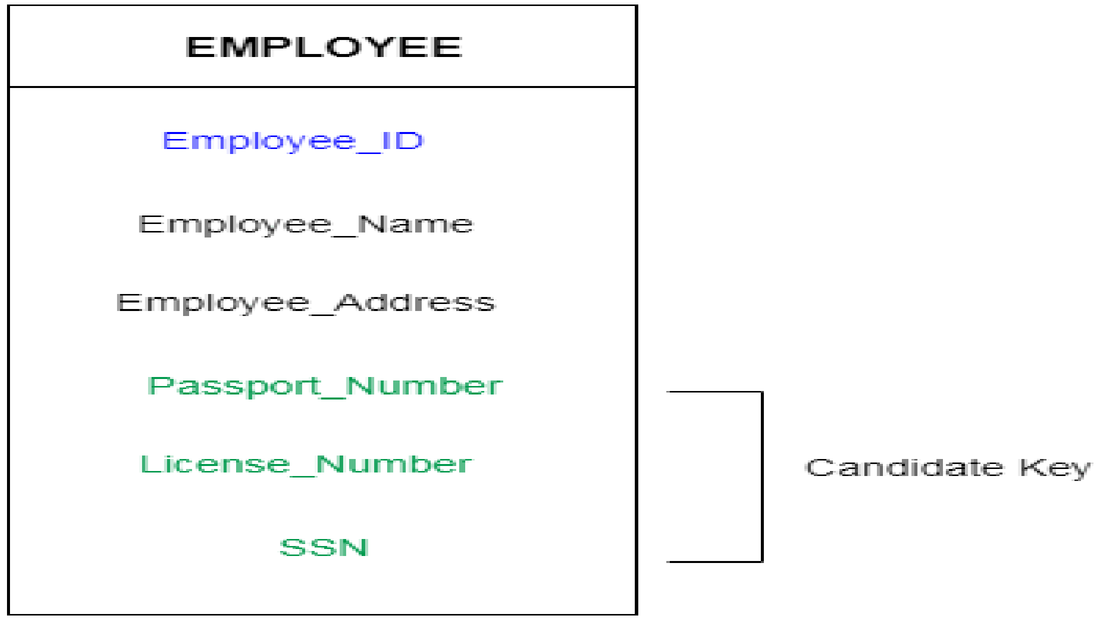
2. Candidate key

- A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
- The remaining attributes except for primary key are considered as 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. Rest of the attributes like SSN, Passport_Number, and License_Number, etc. are considered as a candidate key.
-



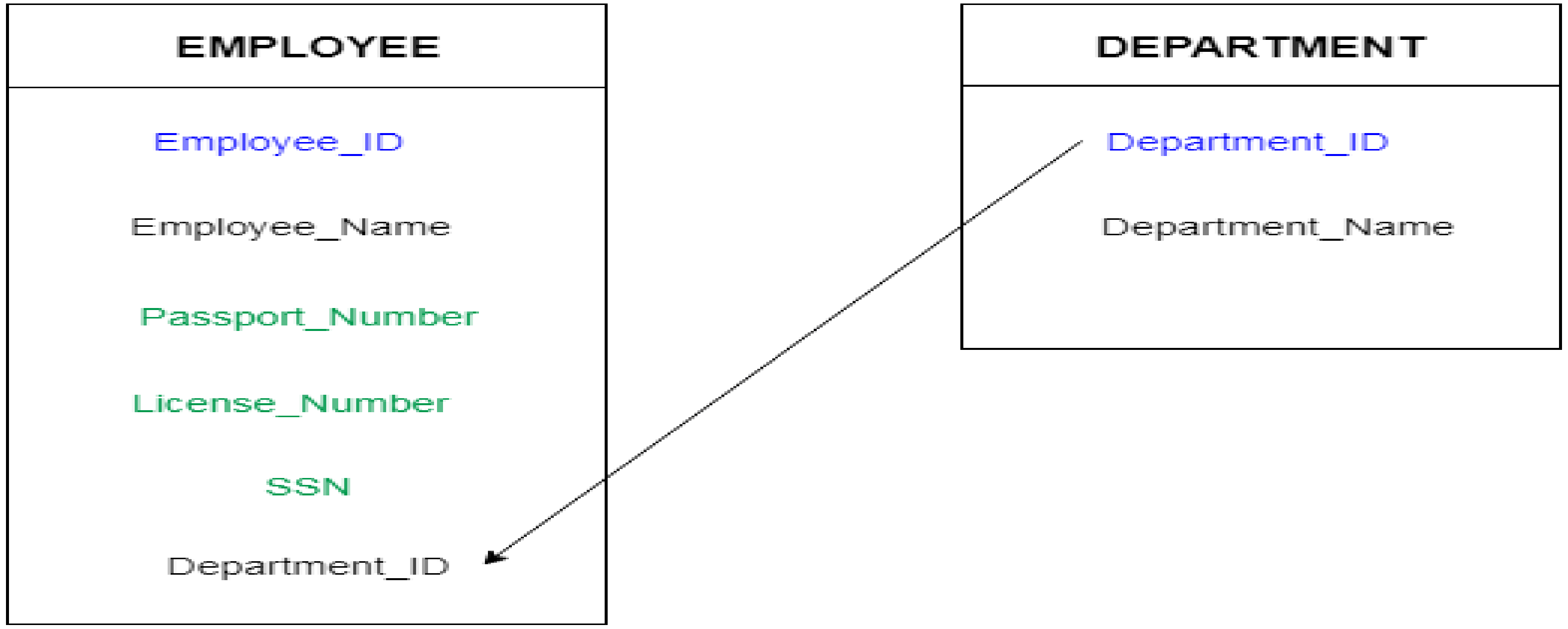
3. Super Key

- Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.
- **For example:** In the EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.
- The super key would be EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME), etc.



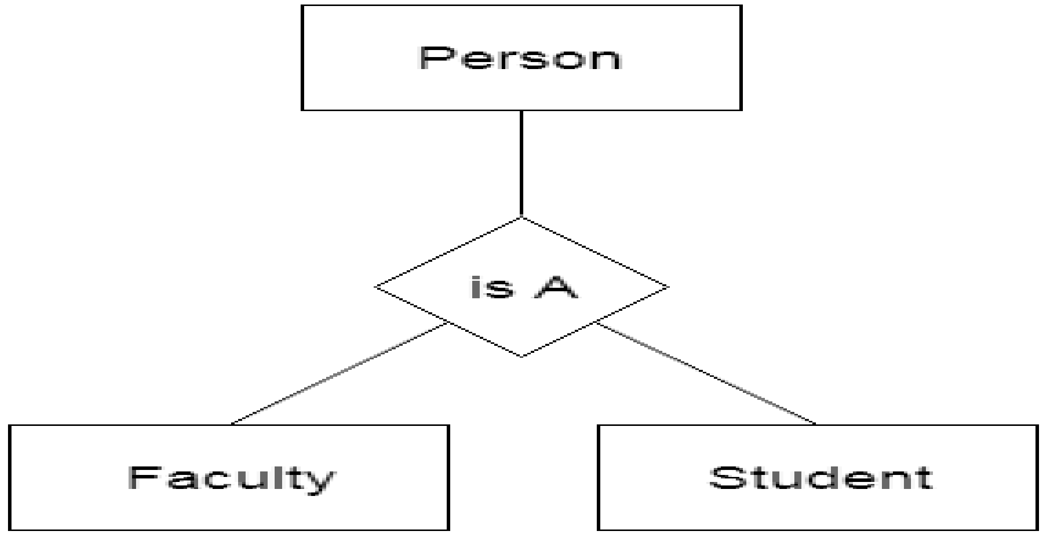
4. Foreign key

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



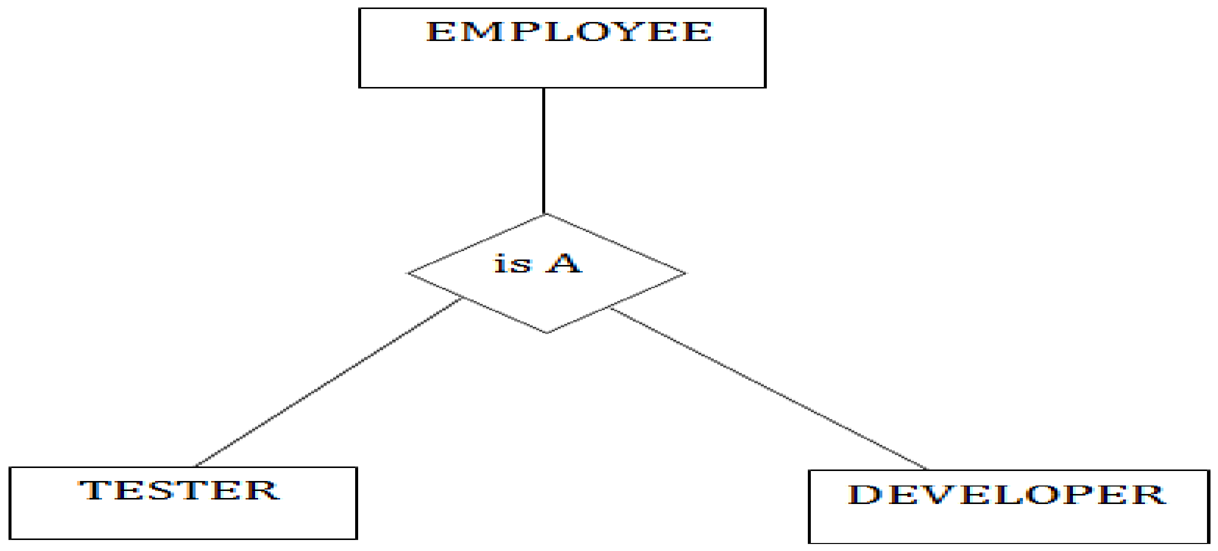
Generalization

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



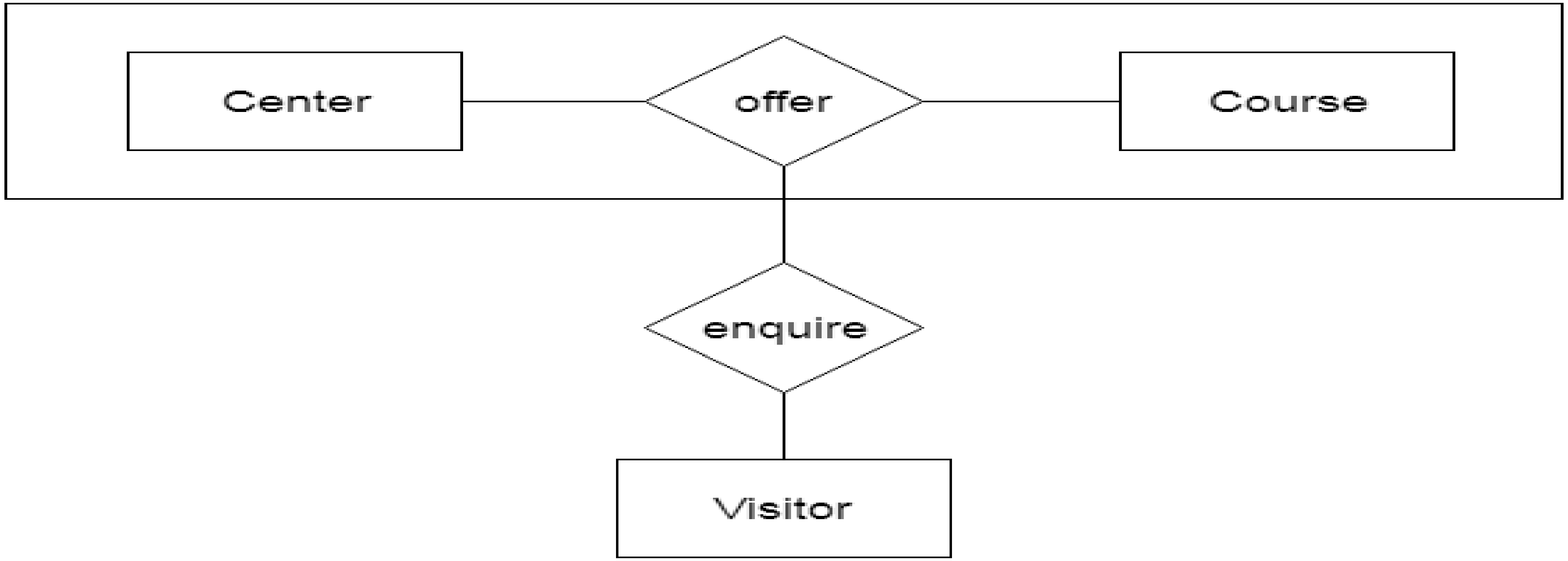
Specialization

- Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
- Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.
- **For example:** In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



Aggregation

- In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.
- **For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



ER diagram to Table

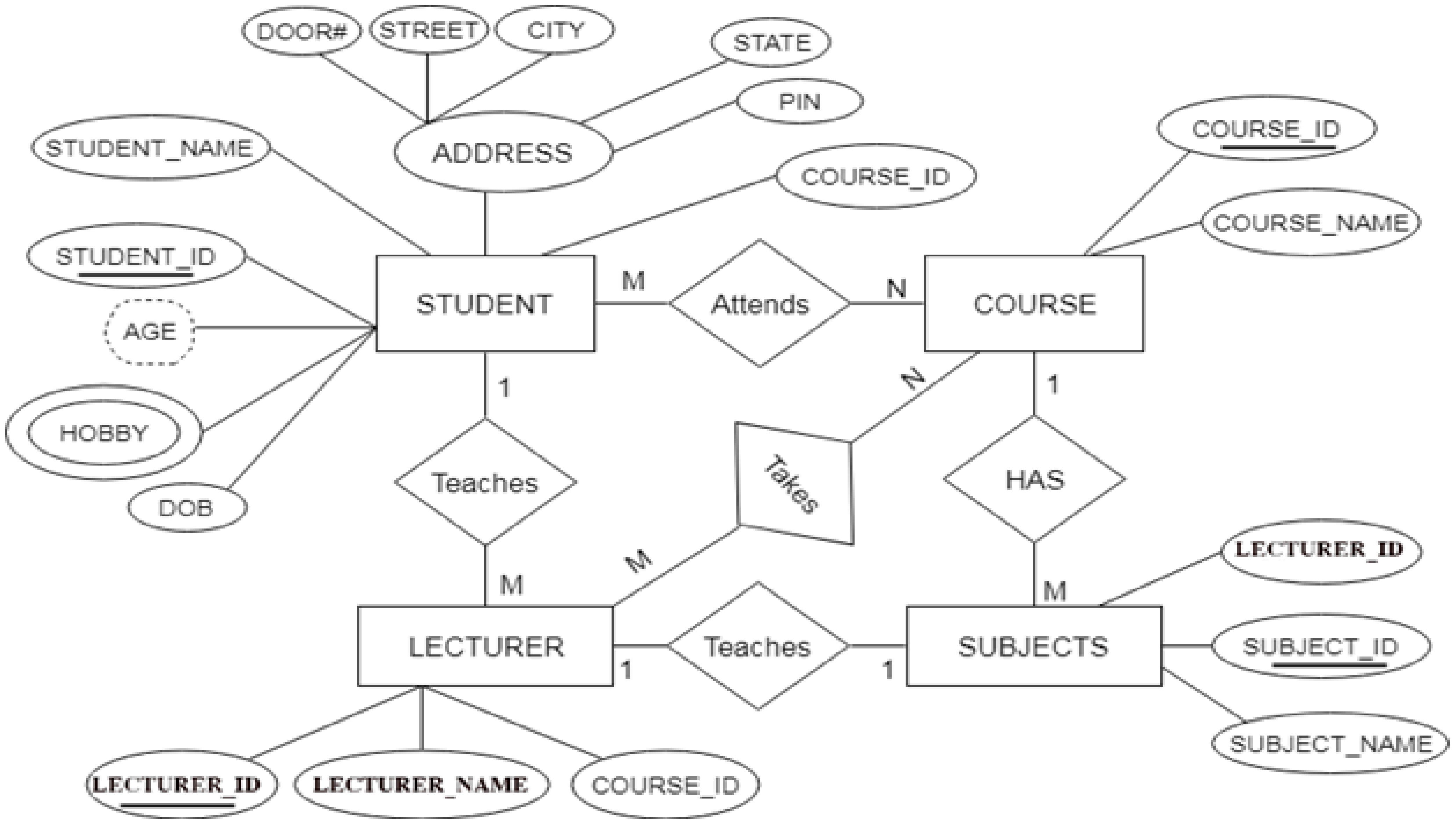
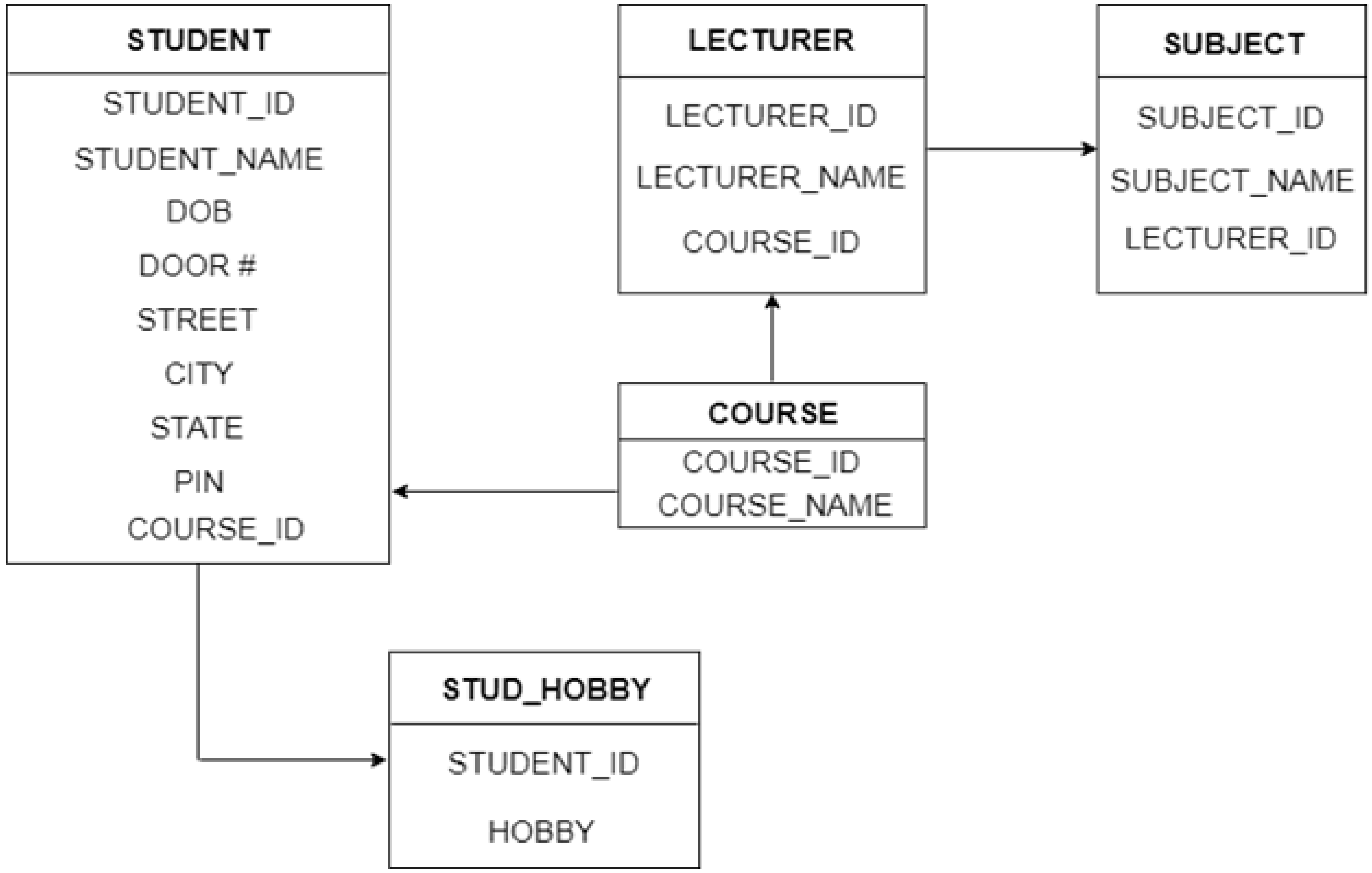


Table structure



DBMS-Unit-2

TOPIC

Relational model:

Relational Model concept

Relational Algebra

Join Operations

Integrity Constraints

Relational Calculus

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 $A \rightarrow B$, 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

- 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
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.
- 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

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

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

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.

1. $X \rightarrow Y$

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

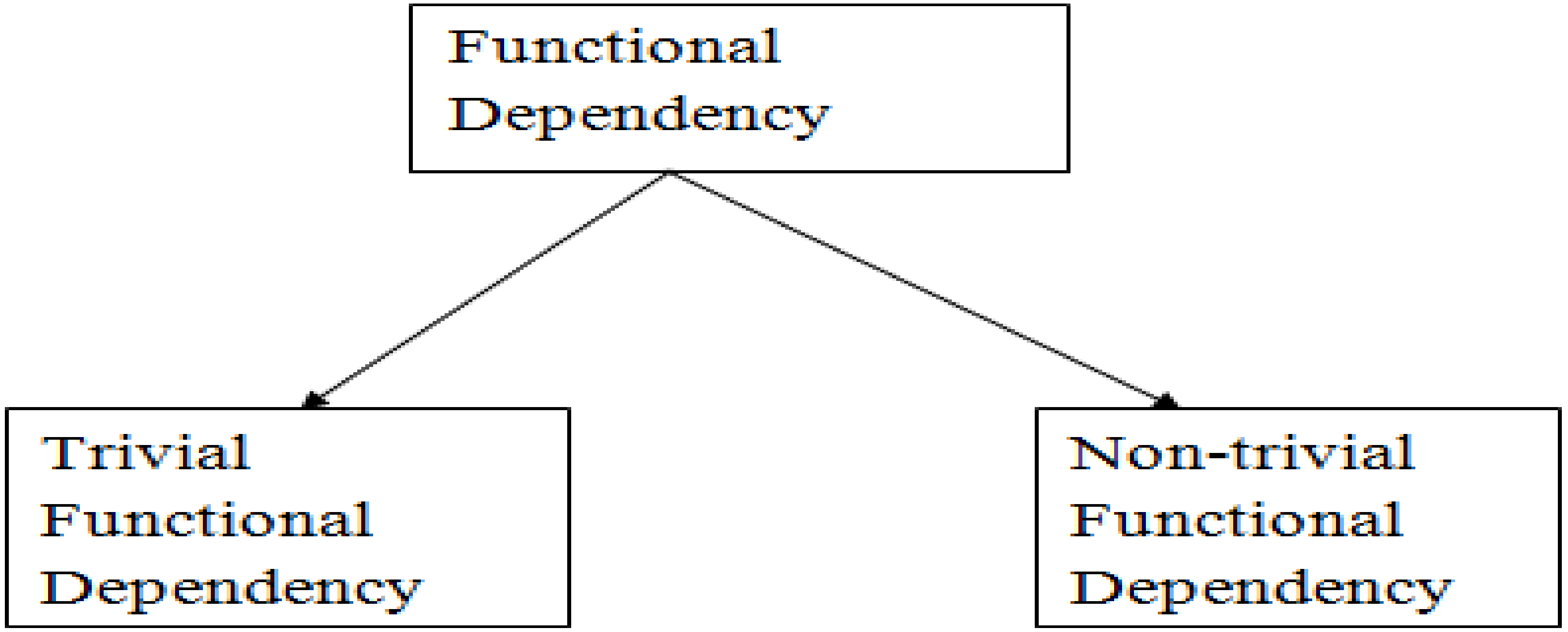
For example:

- 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:

1. Emp_Id $\rightarrow$ Emp_Name

- We can say that Emp_Name is functionally dependent on Emp_Id.

Types of Functional dependency


Functional Dependency
Trivial Functional Dependency
Non-trivial Functional Dependency

1. Trivial functional dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B
- **Example:**

1. Consider a table with two columns Employee_Id and Employee _Name.
2. {Employee_id, Employee_Name} → Employee_Id is a trivia l functional dependency as
3. Employee_Id is a subset of {Employee_Id, Employee_Name}.

4. Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial dependenc

2. Non-trivial functional dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.
- **Example:**

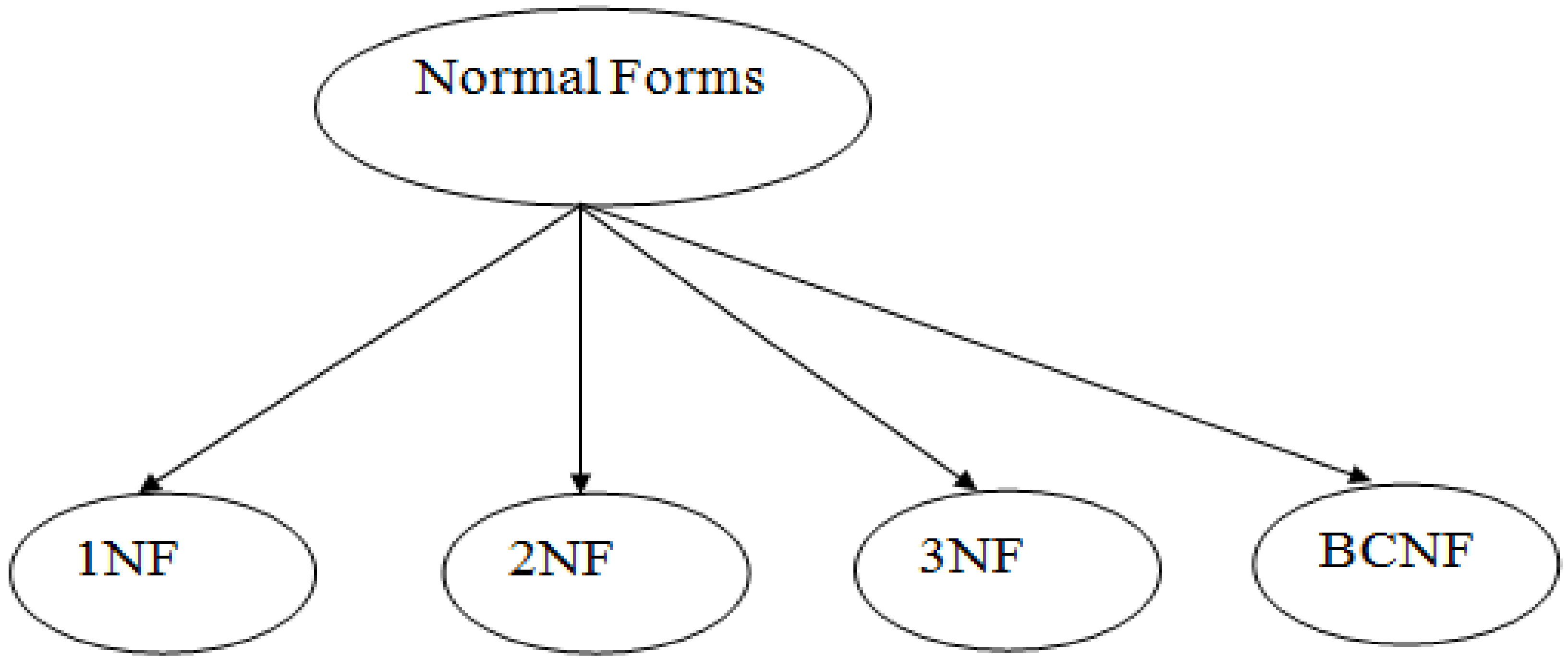
1. ID → Name,
2. Name → DOB

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 the undesirable characteristics like Insertion, Update and Deletion .
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

Types of Normal Forms

- There are the four types of normal forms:



Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

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
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

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:**

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

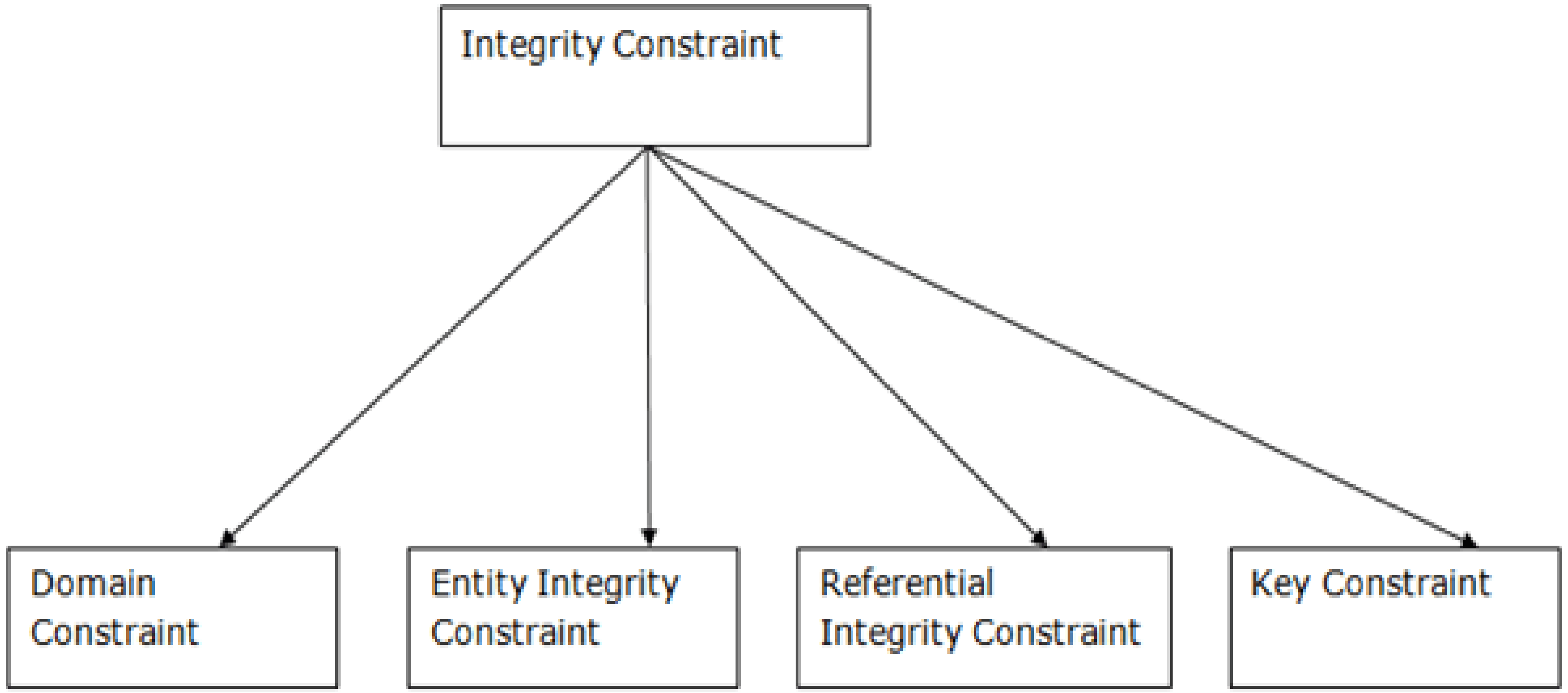
TEACHER_SUBJECT table:

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

Integrity Constraints

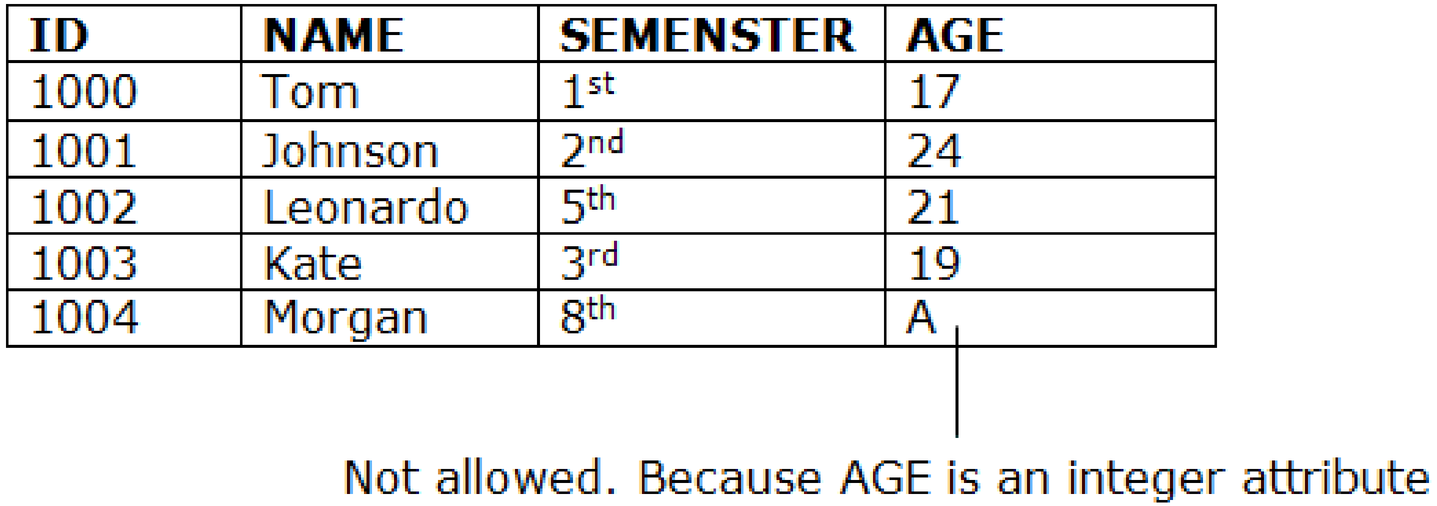
- Integrity constraints are a set of rules.
- It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard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

Types of Integrity Constraint



1. Domain constraints

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



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

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

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

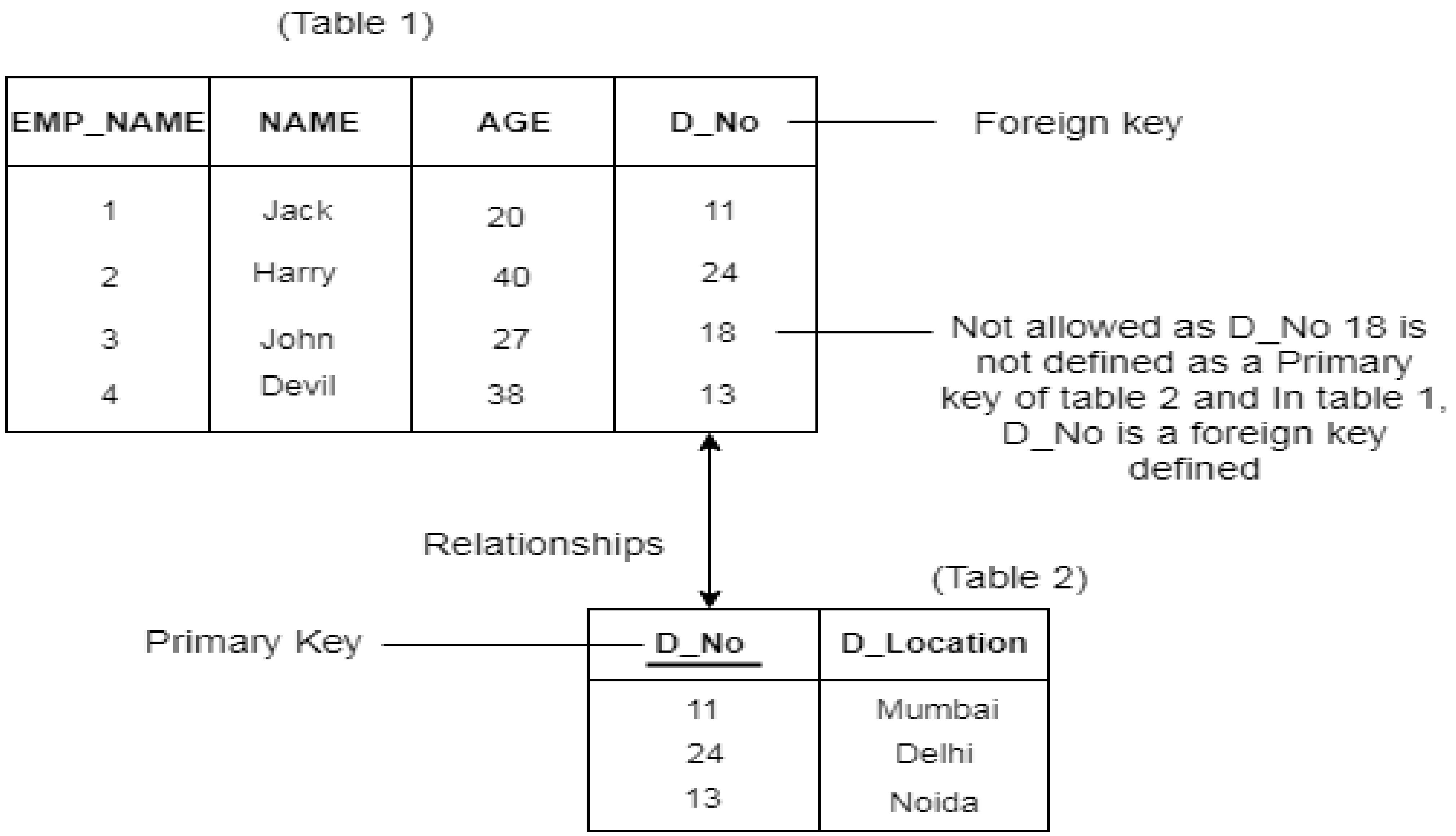
EMPLOYEE

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

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

3. Referential Integrity Constraints

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



(Table 1)

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

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

(Table 2)

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

4. Key constraints

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

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

Not allowed. Because all row must be unique

Introduction of DBMS(Database Management System)

- Database management system is software that is used to manage the database.
- Our DBMS Tutorial includes all topics of DBMS such as introduction, ER model, keys, relational model, join operation, SQL, functional dependency, transaction, concurrency control, etc.
- **DBMS** was designed to solve the fundamental problems associated with storing, managing, accessing, securing, and auditing data in traditional file systems

Who introduced DBMS

- Charles W. Bachman
- In 1960, Charles W. Bachman designed the Integrated **Database System**, the “first” **DBMS**.

Why do we need DBMS

- Proper **database management systems** help increase organizational accessibility to data, which in turn helps the end users share the data quickly and effectively across the organization.
- A **management system** helps get quick solutions to **database** queries, thus making data access faster and more accurate.

Advantages of Database Management System

- Reducing Data Redundancy. The file based data management systems contained multiple files that were stored in many different locations in a system or even across multiple systems. ...
- Sharing of Data. ...
- Data Integrity. ...
- **Data Security**. ...
- Privacy. ...
- Backup and Recovery. ...
- Data **Consistency**.

Join Operations:

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

Example:

EMPLOYEE

EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

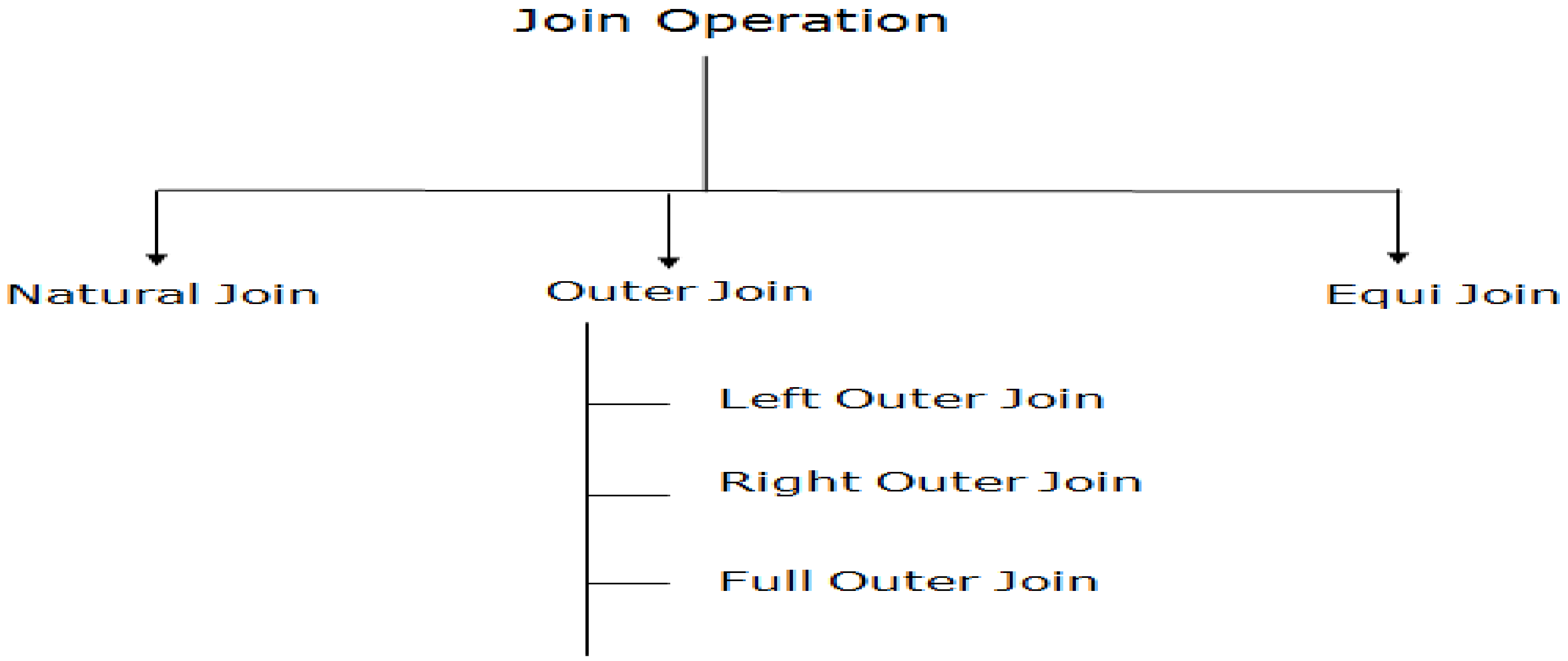
MP_CODE	SALARY
101	50000
102	30000
103	25000

- Operation: (EMPLOYEE ⋈ SALARY)

Result:

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

Types of Join operations:



1. Natural Join:

- A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.
- It is denoted by ⋈.
- **Example:** Let's use the above EMPLOYEE table and SALARY table:

- **Input:**
- $\prod_{EMP\_NAME, SALARY}$ (EMPLOYEE ⋈ SALARY)
- **Output:**
-

EMP_NAME	SALARY
Stephan	50000
Jack	30000
Harry	25000

2. Outer Join:

- The outer join operation is an extension of the join operation.
- It is used to deal with missing information.
- **Example:**
- **EMPLOYEE**

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

FACT_WORKERS

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

- **Input:**
- EMPLOYEE $\bowtie$ FACT_WORKERS
- **Output:**

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

An outer join is basically of three types:

a. Left outer join

b. Right outer join

c. Full outer join

Left outer join:

- Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In the left outer join, tuples in R have no matching tuples in S.
- It is denoted by ⟕.
- **Example:** Using the above EMPLOYEE table and FACT_WORKERS table
- **Input:**

EMPLOYEE $\bowtie$ FACT_WORKERS

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

b. Right outer join:

- Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In right outer join, tuples in S have no matching tuples in R.
- It is denoted by ⟖.
- **Example:** Using the above EMPLOYEE table and FACT_WORKERS Relation
- **Input:**
- EMPLOYEE ⟖ FACT_WORKERS

Output:

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

c. Full outer join:

- Full outer join is like a left or right join except that it contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.
- **Example:** Using the above EMPLOYEE table and FACT_WORKERS table
- **Input:**
- EMPLOYEE ⟗ FACT_WORKERS

Output:

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

3. Equi join:

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

CUSTOMER RELATION

CLASS_ID	NAME
1	John
2	Harry
3	Jackson

PRODUCT_ID	CITY
1	Delhi
2	Mumbai
3	Noida

Input:

- CUSTOMER ⋈ PRODUCT
- **Output:**

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

Join Operations:

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

Example:
EMPLOYEE

EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

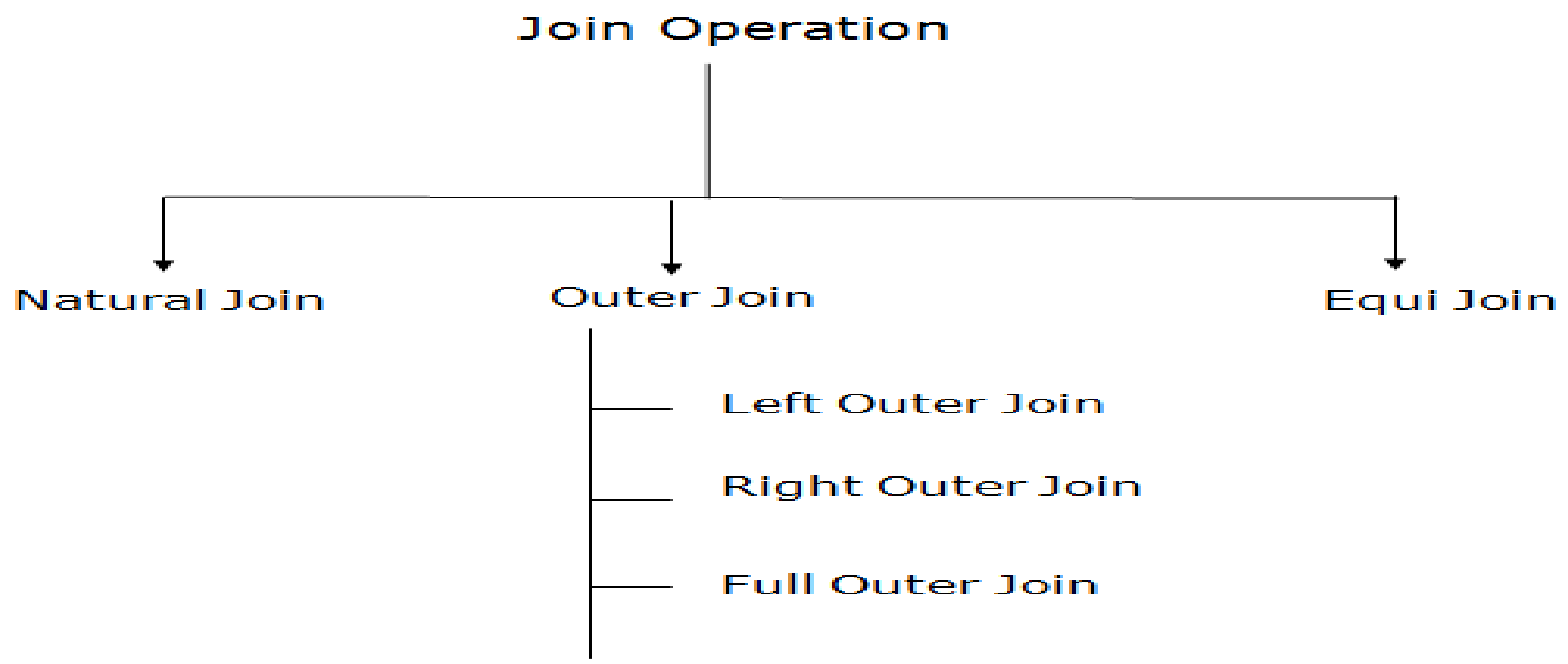
MP_CODE	SALARY
101	50000
102	30000
103	25000

- Operation: $(\text{EMPLOYEE} \bowtie \text{SALARY})$

Result:

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

Types of Join operations:



1. Natural Join:

- A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.
- It is denoted by ⋈.
- **Example:** Let's use the above EMPLOYEE table and SALARY table:

- **Input:**
- $\prod_{EMP\_NAME, SALARY}$ (EMPLOYEE ⋈ SALARY)
- **Output:**
-

EMP_NAME	SALARY
Stephan	50000
Jack	30000
Harry	25000

2. Outer Join:

- The outer join operation is an extension of the join operation.
- It is used to deal with missing information.
- **Example:**
- **EMPLOYEE**

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

FACT_WORKERS

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

- **Input:**
- EMPLOYEE ⋈ FACT_WORKERS
- **Output:**

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

An outer join is basically of three types:

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

Left outer join:

- Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In the left outer join, tuples in R have no matching tuples in S.
- It is denoted by ⟕.
- **Example:** Using the above EMPLOYEE table and FACT_WORKERS table
- **Input:**

EMPLOYEE $\bowtie$ FACT_WORKERS

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

b. Right outer join:

- Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In right outer join, tuples in S have no matching tuples in R.
- It is denoted by ⟖.
- **Example:** Using the above EMPLOYEE table and FACT_WORKERS Relation
- **Input:**
- EMPLOYEE ⟖ FACT_WORKERS

Output:

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

c. Full outer join:

- Full outer join is like a left or right join except that it contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.
- **Example:** Using the above EMPLOYEE table and FACT_WORKERS table
- **Input:**
- EMPLOYEE ⟗ FACT_WORKERS

Output:

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

3. Equi join:

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

CUSTOMER RELATION

CLASS_ID	NAME
1	John
2	Harry
3	Jackson

PRODUCT_ID	CITY
1	Delhi
2	Mumbai
3	Noida

Input:

- CUSTOMER ⋈ PRODUCT
- **Output:**

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

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 $X \rightarrow Y$.

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

Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF.
- It is stricter than 3NF.
- A table is in BCNF if every functional dependency $X \to Y$, 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 → EMP_COUNTRY
2. EMP_DEPT → {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
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

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.

Relational Model concept

- 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 must have a domain.
- **Relational instance:** In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples.
- **Relational schema:** A relational schema contains the name of the relation and name of all columns or attributes.
- **Relational key:** In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

Example: STUDENT Relation

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

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

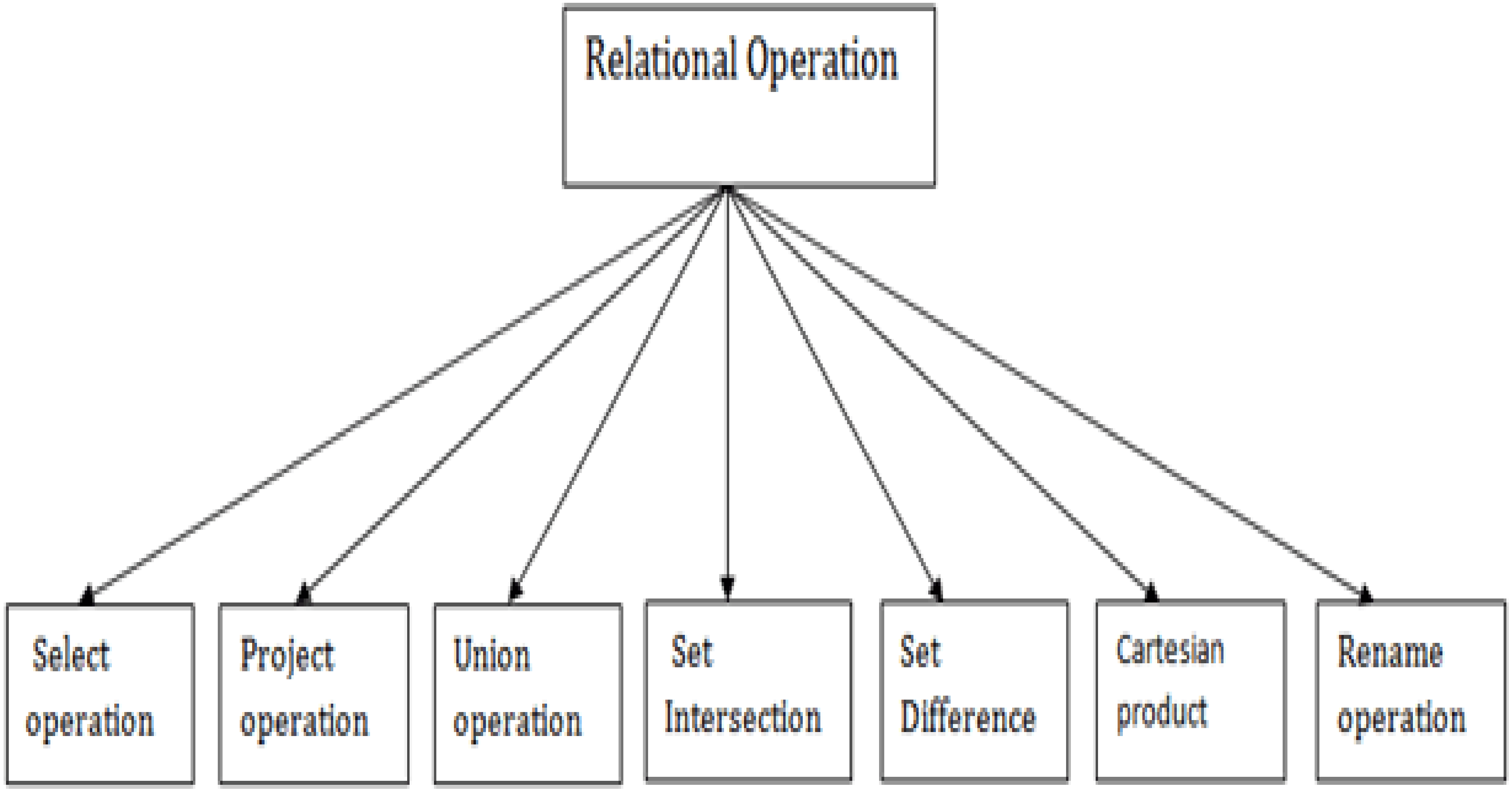
Properties of Relations

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

Relational Algebra

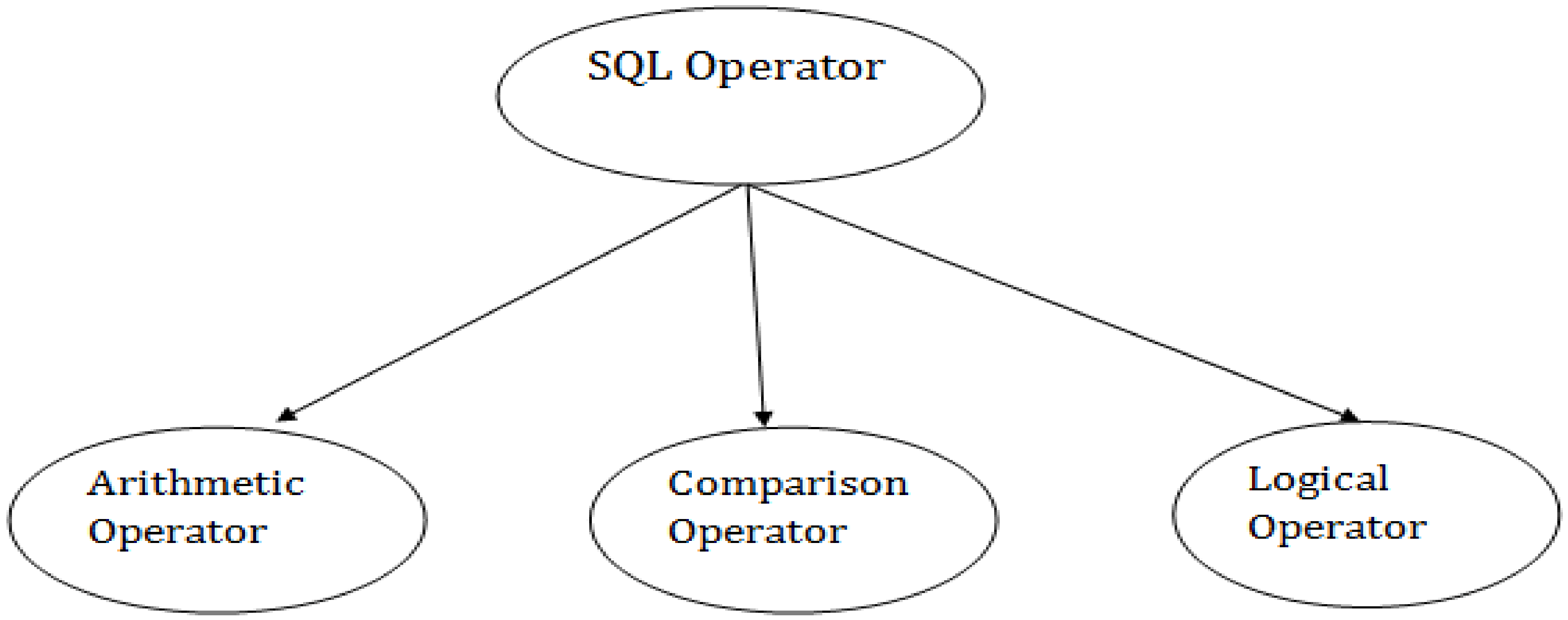
- Relational algebra is a procedural query language.
- It gives a step by step process to obtain the result of the query.
- It uses operators to perform queries.

Types of Relational operation



SQL Operator

- There are various types of SQL operator:
-



SQL Arithmetic Operators

- Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.
-

Operator	Description	Example
+	It adds the value of both operands.	a+b will give 30
-	It is used to subtract the right-hand operand from the left-hand operand.	a-b will give 10
*	It is used to multiply the value of both operands.	a*b will give 200
/	It is used to divide the left-hand operand by the right-hand operand.	a/b will give 2
%	It is used to divide the left-hand operand by the right-hand operand and returns reminder.	a%b will give 0

SQL Comparison Operators:

- Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10

Operator	Description	Example
=	It checks if two operands values are equal or not, if the values are equal then condition becomes true.	(a=b) is not true
!=	It checks if two operands values are equal or not, if values are not equal, then condition becomes true.	(a!=b) is true
<>	It checks if two operands values are equal or not, if values are not equal then condition becomes true.	(a<>b) is true
>	It checks if the left operand value is greater than right operand value, if yes then condition becomes true.	(a>b) is not true
<	It checks if the left operand value is less than right operand value, if yes then condition becomes true.	(a<b) is true
>=	It checks if the left operand value is greater than or equal to the right operand value, if yes then condition becomes true.	(a>=b) is not true
<=	It checks if the left operand value is less than or equal to the right operand value, if yes then condition becomes true.	(a<=b) is true
!<	It checks if the left operand value is not less than the right operand value, if yes then condition becomes true.	(a!=b) is not true
!>	It checks if the left operand value is not greater than the right operand value, if yes then condition becomes	(a!>b) is true

SQL Logical Operators

- There is the list of logical operator used in SQL:

Operator	Description
ALL	It compares a value to all values in another value set.
AND	It allows the existence of multiple conditions in an SQL statement.
ANY	It compares the values in the list according to the condition.
BETWEEN	It is used to search for values that are within a set of values.
IN	It compares a value to that specified list value.
NOT	It reverses the meaning of any logical operator.
OR	It combines multiple conditions in SQL statements.
EXISTS	It is used to search for the presence of a row in a specified table.
LIKE	It compares a value to similar values using wildcard operator.

SQL - UNIONS CLAUSE

- The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.
- To use this UNION clause, each SELECT statement must have
- The same number of columns selected
- The same number of column expressions
- The same data type and
- Have them in the same order
- But they need not have to be in the same length.

- **Syntax**
- **The basic syntax of** a **UNION** clause is as follows −

- SELECT column1,[column2]
- FROM table1[,table2]
- [where condition]
- UNION
- SSELECT column1[,column2]
- FROM table1[,table2]
- [where condition]
- Here, the given condition could be any given expression based on your requirement.

except_()

- The SQL EXCEPT clause/operator is used to combine two SELECT statements and return rows from the first SELECT statement that are not returned by the second SELECT statement.
- The except_() function generates a SELECT expression with EXCEPT clause.

intersect()

- Using INTERSECT operator, SQL displays common rows from both the SELECT statements.
- The intersect() function implements this behaviour.

SQL

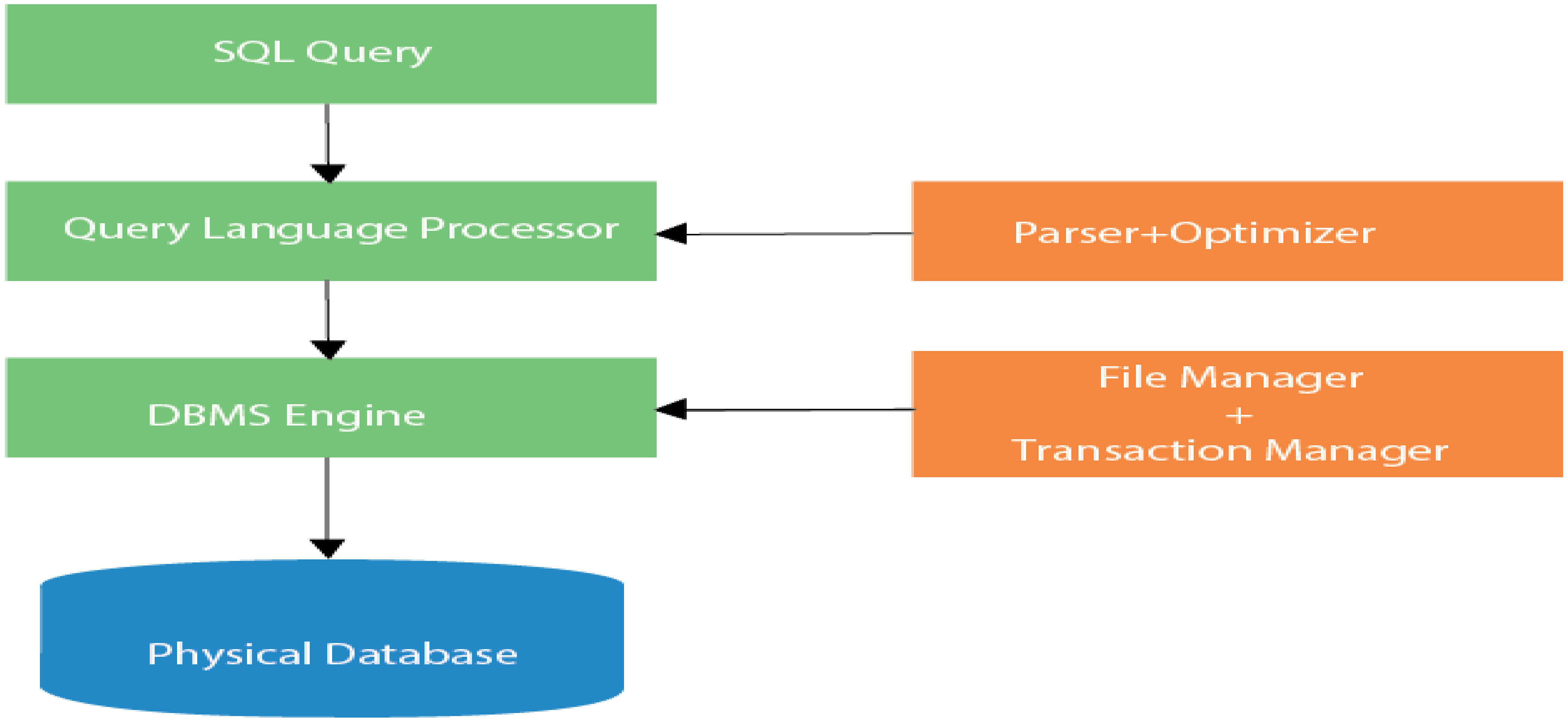
- SQL stands for Structured Query Language.
- It is used for storing and managing data in relational database management system (RDMS).
- It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
- All the RDBMS like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

Rules:

- SQL follows the following rules:
- Structure query language is not case sensitive.
- Generally, keywords of SQL are written in uppercase.
- Statements of SQL are dependent on text lines.
- We can use a single SQL statement on one or multiple text line.
- Using the SQL statements, you can perform most of the actions in a database.
- SQL depends on tuple relational calculus and relational algebra.

SQL process:

- When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine determines that how to interpret the task.
- In the process, various components are included.
- These components can be optimization Engine, Query engine, Query dispatcher, classic, etc.
- All the non-SQL queries are handled by the classic query engine, but SQL query engine won't handle logical files.


SQL Query
Query Language Processor
Parser+Optimizer
DBMS Engine
File Manager
+
Transaction Manager
Physical Database

Characteristics of SQL

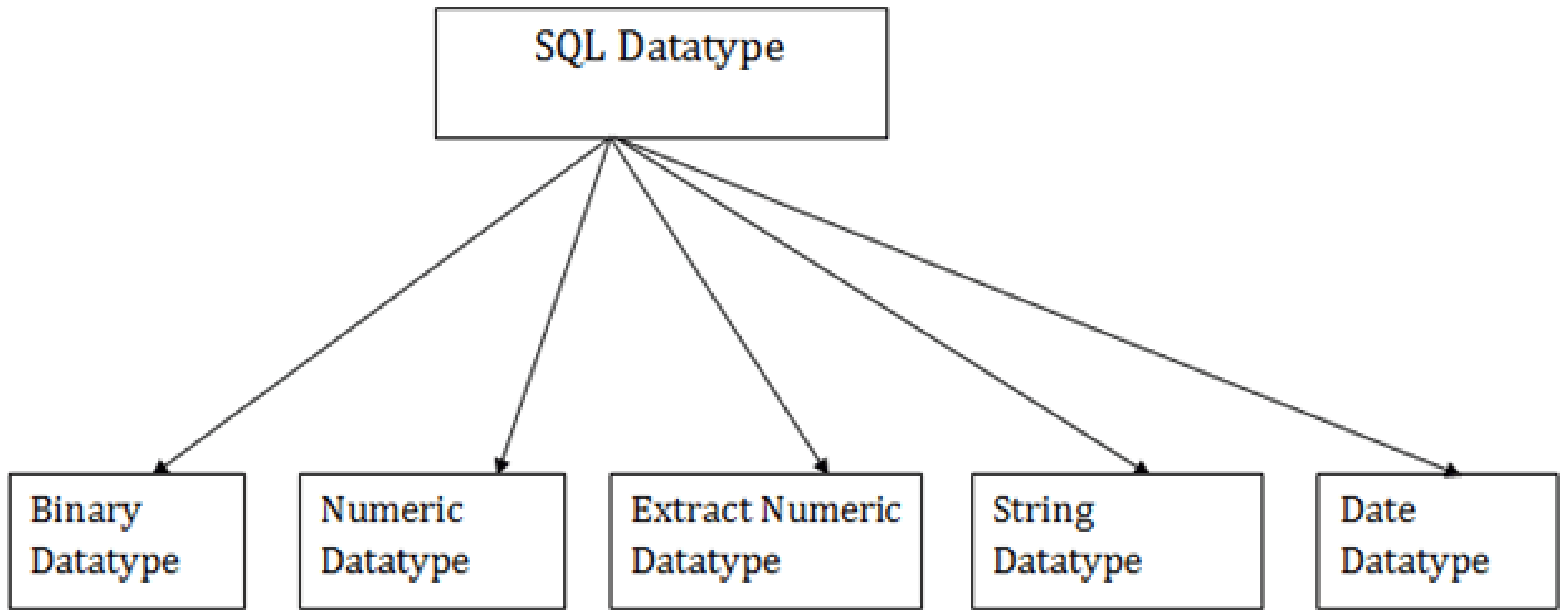
- SQL is easy to learn.
- SQL is used to access data from relational database management systems.
- SQL can execute queries against the database.
- SQL is used to describe the data.
- SQL is used to define the data in the database and manipulate it when needed.
- SQL is used to create and drop the database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

Advantages of SQL

- There are the following advantages of SQL:
- High speed
- Using the SQL queries, the user can quickly and efficiently retrieve a large amount of records from a database.
- No coding needed
- In the standard SQL, it is very easy to manage the database system. It doesn't require a substantial amount of code to manage the database system.
- Well defined standards
- Long established are used by the SQL databases that are being used by ISO and ANSI.
- Portability
- SQL can be used in laptop, PCs, server and even some mobile phones.
- Interactive language
- SQL is a domain language used to communicate with the database. It is also used to receive answers to the complex questions in seconds.
- Multiple data view
- Using the SQL language, the users can make different views of the database structure.

SQL Datatype

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


SQL Datatype
Binary Datatype
Numeric Datatype
Extract Numeric Datatype
String Datatype
Date Datatype

1. Binary Datatypes

Data Type	Description
binary	It has a maximum length of 8000 bytes. It contains fixed-length binary data.
varbinary	It has a maximum length of 8000 bytes. It contains variable-length binary data.
image	It has a maximum length of 2,147,483,647 bytes. It contains variable-length binary data.

2. Approximate Numeric Datatype :

Data type	From	To	Description
float	-1.79E + 308	1.79E + 308	It is used to specify a floating-point value e.g. 6.2, 2.9 etc.
real	-3.40e + 38	3.40E + 38	It specifies a single precision floating point number

3. Exact Numeric Datatype

Data type	Description
int	It is used to specify an integer value.
smallint	It is used to specify small integer value.
bit	It has the number of bits to store.
decimal	It specifies a numeric value that can have a decimal number.
numeric	It is used to specify a numeric value.

4. Character String Datatype

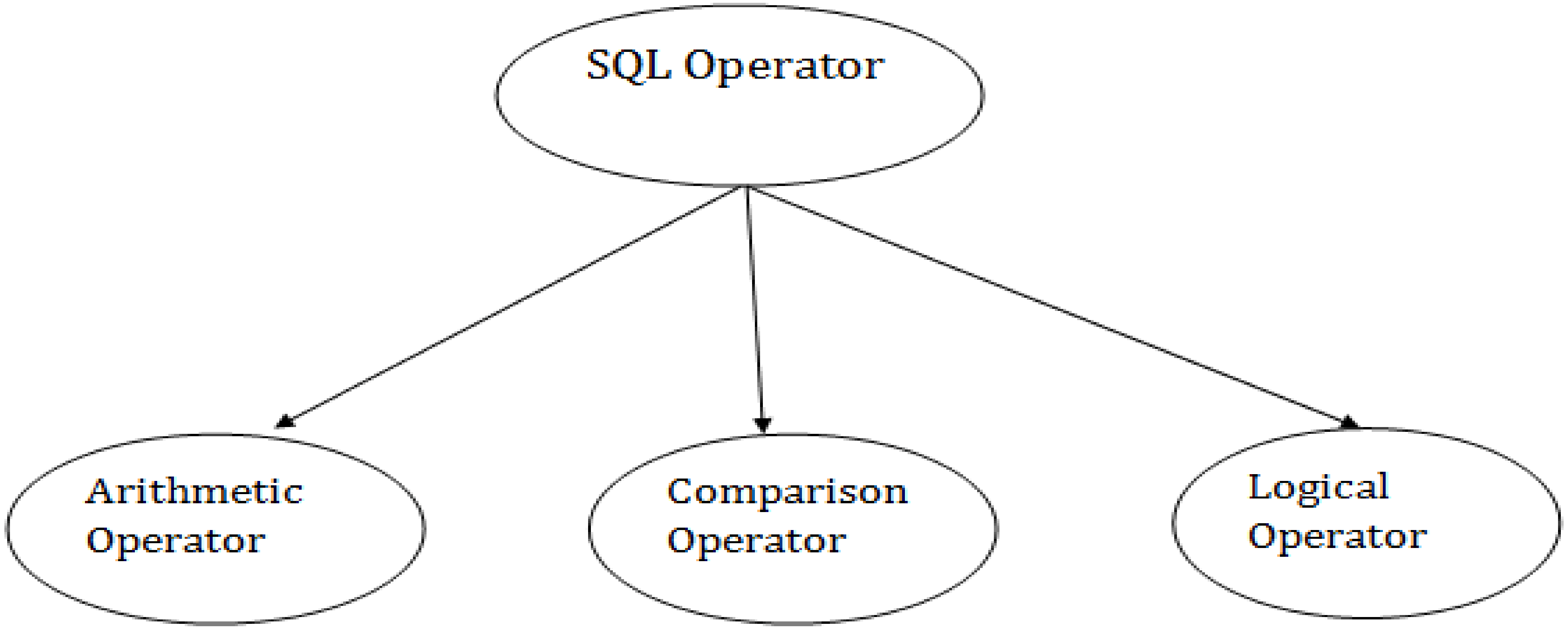
Data type	Description
char	It has a maximum length of 8000 characters. It contains Fixed-length non-unicode characters.
varchar	It has a maximum length of 8000 characters. It contains variable-length non-unicode characters.
text	It has a maximum length of 2,147,483,647 characters. It contains variable-length non-unicode characters.

5. Date and time Datatypes

Datatype	Description
date	It is used to store the year, month, and days value.
time	It is used to store the hour, minute, and second values.
timestamp	It stores the year, month, day, hour, minute, and the second value.

SQL Commands

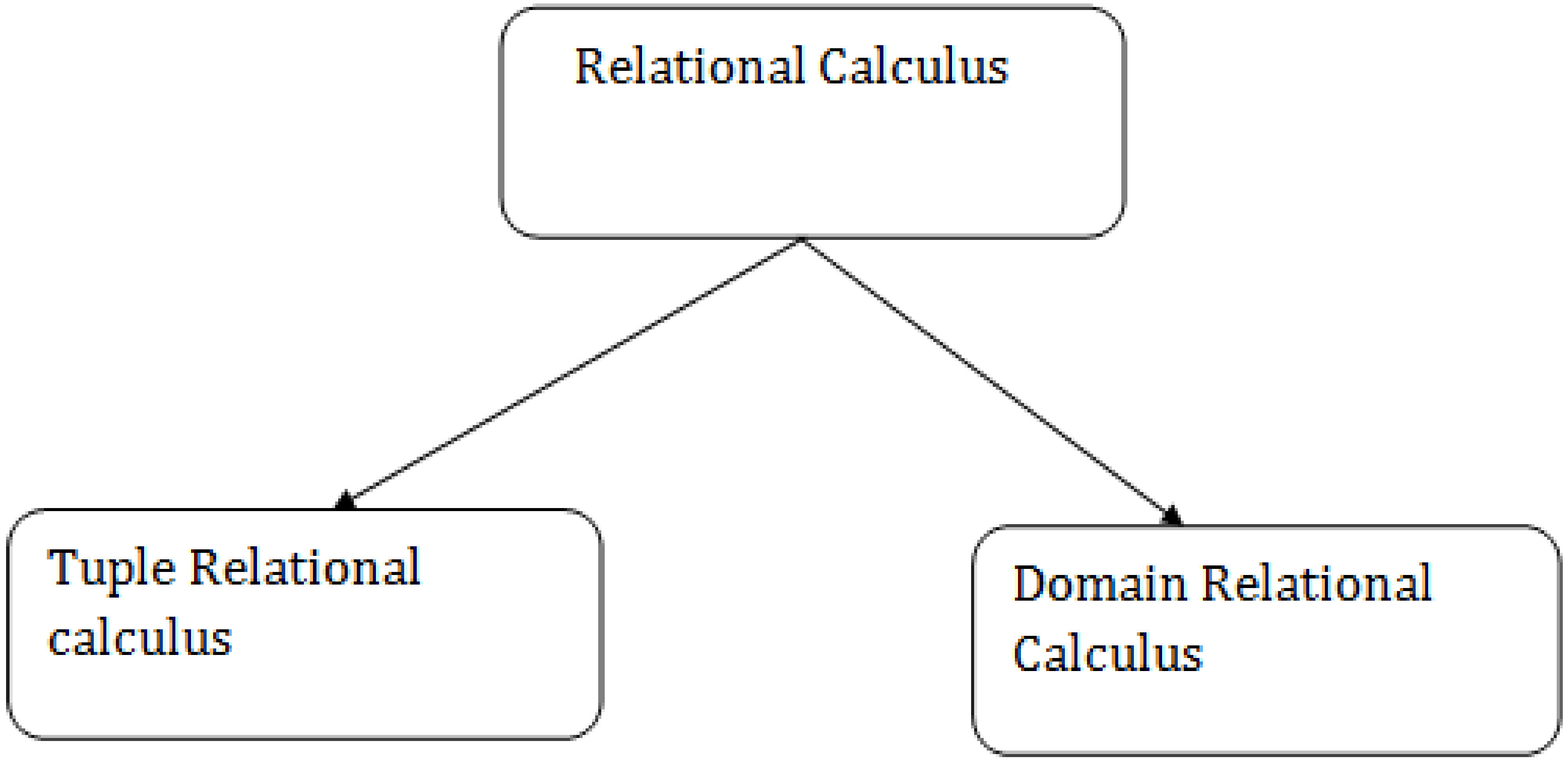
SQL Operator



Relational Calculus

- Relational calculus is a non-procedural query language.
- In the non-procedural query language, the user is concerned with the details of how to obtain the end results.
- The relational calculus tells what to do but never explains how to do.

Types of Relational calculus:



1. Tuple Relational Calculus (TRC)

- The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation.
- The result of the relation can have one or more tuples.
- **Notation:**

1. {T | P (T)}   or {T | Condition (T)}

- Where
- **T** is the resulting tuples
- **P(T)** is the condition used to fetch T.

For example

1. { T.name | Author(T) AND T.article = 'database' }

- **OUTPUT:** This query selects the tuples from the AUTHOR relation.
- It returns a tuple with 'name' from Author who has written an article on 'database'.

2. Domain Relational Calculus (DRC)

- The second form of relation is known as Domain relational calculus.
- In domain relational calculus, filtering variable uses the domain of attributes.
- Domain relational calculus uses the same operators as tuple calculus.
- It uses logical connectives ∧ (and), ∨ (or) and ┓ (not).
- It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable.

- **Notation:**

1. { a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

- Where
- **a1, a2** are attributes
  P stands for formula built by inner attributes
- **For example:**

1. {< article, page, subject > | ∈ DBMS ∧ subject = 'database'}

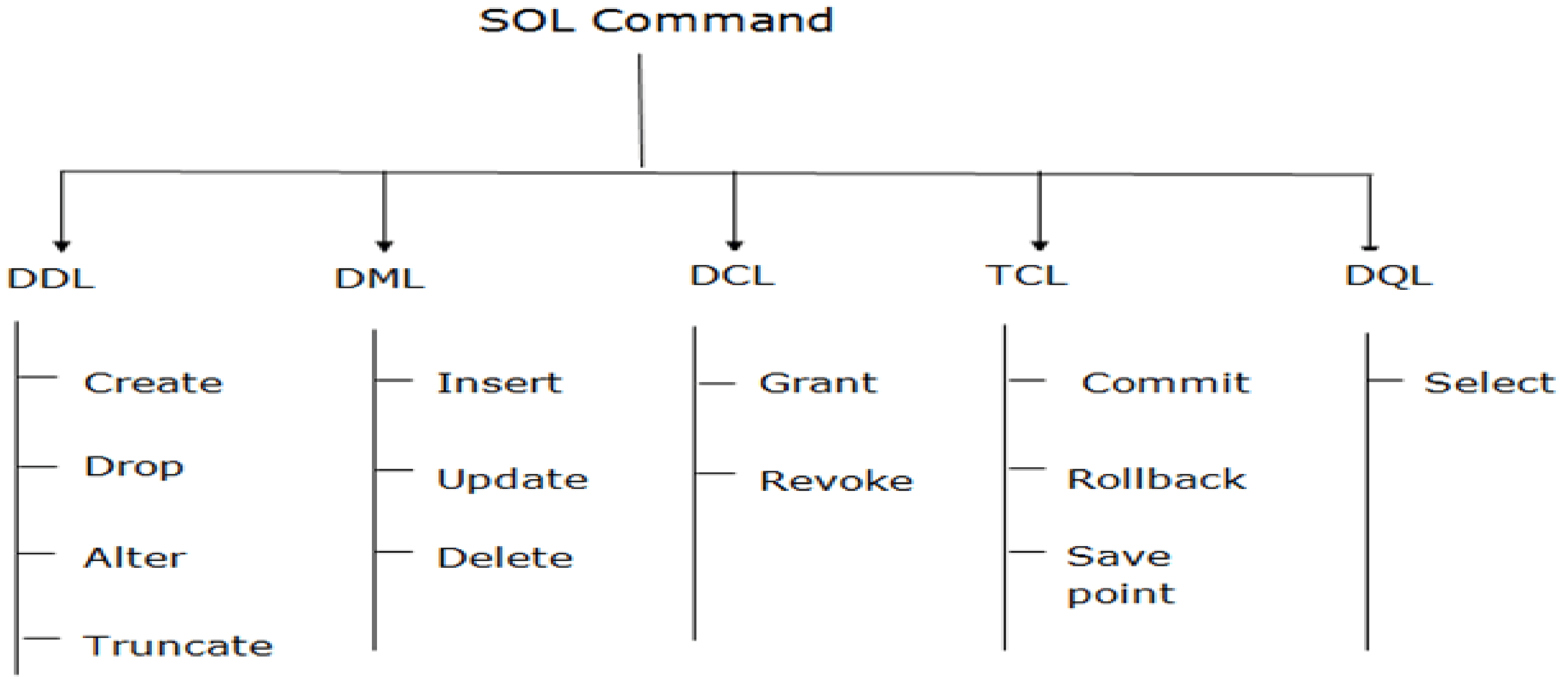
- **Output:** This query will yield the article, page, and subject from the relational DBMS, where the subject is a database.

SQL Commands

- 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.
- 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

- There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- 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:
- CREATE
- ALTER
- DROP
- TRUNCATE
- **a. 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), DOB DATE);
- **b. DROP:** It is used to delete both the structure and record stored in the table.
- **Syntax**
- DROP TABLE ;
- **Example**
- DROP TABLE EMPLOYEE;

c. 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

1. ALTER TABLE table_name ADD column_name COLUMN-definition;

- To modify existing column in the table:

1. ALTER TABLE MODIFY(COLUMN DEFINITION....);

- **EXAMPLE**

1. ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));
2. ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

d. TRUNCATE

- It is used to delete all the rows from the table and free the space containing the table.
- **Syntax:**

1. TRUNCATE TABLE table_name;

- **Example:**

1. TRUNCATE TABLE EMPLOYEE;

2. Data Manipulation Language

- DML commands are used to modify the database.
- It is responsible for all form of changes in the database.
- 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:
- INSERT
- UPDATE
- 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
2. (col1, col2, col3,.... col N)
3. VALUES (value1, value2, value3, ... valueN);

- Or

1. INSERT INTO TABLE_NAME
2. VALUES (value1, value2, value3, ... valueN);

- **For example:**

1. INSERT INTO Database (Author, Subject) VALUES ("Sonoo", "DBMS");

b. UPDATE:

- **Syntax:**

1. UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]

- **For example:**

```
1.UPDATE students
2.SET User_Name = 'Sonoo'
3.WHERE Student_Id = '3'
```

c. DELETE:

- It is used to remove one or more row from a table.
- **Syntax:**

1. DELETE FROM table_name [WHERE condition];

- **For example:**

1. DELETE FROM javatpoint
2. WHERE Author="Sonoo";

3. Data Control Language

- DCL commands are used to grant and take back authority from any database user.
- Here are some commands that come under DCL:
- Grant
- Revoke

a. Grant

- It is used to give user access privileges to a database.
- **Example**

1. GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

b. Revoke:

- It is used to take back permissions from the user.
- **Example**

1. REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

4. Transaction Control Language

- TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.
- These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.
- Here are some commands that come under TCL:
- COMMIT
- ROLLBACK
- SAVEPOINT

a. Commit:

- Commit command is used to save all the transactions to the database.
- **Syntax:**

```
1.COMMIT;
```

- **Example:**

```
1.DELETE FROM CUSTOMERS
2.WHERE AGE = 25;
3.COMMIT;
```

b. Rollback

- Rollback command is used to undo transactions that have not already been saved to the database.
- **Syntax:**

```
1.ROLLBACK;
```

- **Example:**

```
1.DELETE FROM CUSTOMERS
2.WHERE AGE = 25;
3.ROLLBACK;
```

c. SAVEPOINT

- It is used to roll the transaction back to a certain point without rolling back the entire transaction.
- **Syntax:**

1. SAVEPOINT SAVEPOINT_NAME;

5. Data Query Language

- DQL is used to fetch the data from the database.
- It uses only one command:
- SELECT

a. SELECT:

- This is the same as the projection operation of relational algebra.
- It is used to select the attribute based on the condition described by WHERE clause.
- **Syntax:**

1. SELECT expressions
2. FROM TABLES
3. WHERE conditions;

- **For example:**

1. SELECT emp_name
2. FROM employee
3. WHERE age > 20;

Triggers

- A trigger is a set of actions, which are performed for responding to an INSERT, UPDATE or DELETE operation on a specified table in the database.
- Triggers are stored in the database at once.
- They handle governance of data.
- They can be accessed and shared among multiple applications.
- The advantage of using triggers is, if any change needs to be done in the application, it is done at the trigger; instead of changing each application that is accessing the trigger.
- Triggers are easy to maintain and they enforce faster application development. Triggers are defined using an SQL statement “CREATE TRIGGER”.

Types of triggers

- There are two types of triggers:
- 1. BEFORE triggers
- They are executed before any SQL operation.
- 2. AFTER triggers
- They are executed after any SQL operation.

Creating a BEFORE trigger

Let us see how to create a sequence of trigger:

Syntax:

```
 create sequence <seq_name>
```

Example: Creating a sequence of triggers for table shopper.sales1

```
 create sequence sales1_seq as int start with 1 increment by 1
```

Syntax:

```
 create trigger <trigger_name> no cascade before insert on
 <table_name> referencing new as <table_object>
for each row set <table_object>.<col_name>=nextval for <sequence_name>
```

Example:

```
 insert into shopper.sales1
(itemname, qty, price) values('bicks', 100, 24.00)
```

Retrieving values from table

- Let us see how to retrieve values from a table:
- **Syntax:**
- Select * from<tablename>
- **Example**:
- Select * from shopper.sales1

Output:

```
ID ITEMNAME QTY
3  bicks    100
2  bread     100
```

Creating an AFTER trigger

Syntax:

```
 create trigger <trigger_name> no cascade before insert on <table_name>
 referencing new as <table_object>
for each row set <table_object>.<col_name>=nextval for <sequence_name>
```

Example: [To insert and retrieve the values]

```
 create trigger sales1_tri_after after insert on
shopper.sales1 for
 each row mode sql begin atomic update
shopper.sales1 set price=qty*price; end
```

Output:

```
insert into shopper.sales1(itemname,qty,price)
values('chiken',100,124.00)
```

output

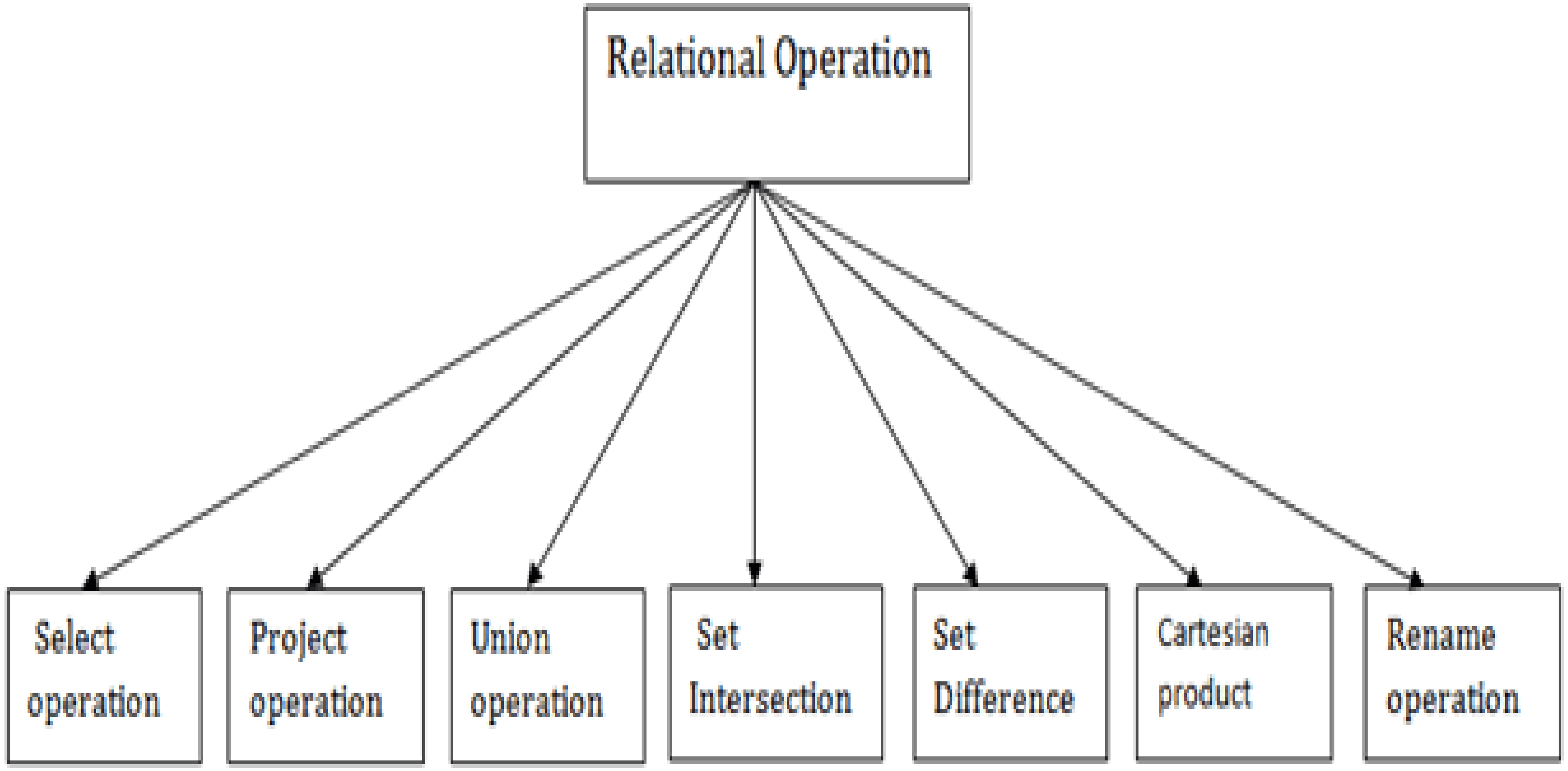
```
ID ITEMNAME QTY PRICE
3  bicks     100 2400.00
 4  chiken  100 12400.00
 2  bread   100 2400.
```

```
db2 drop trigger slaes1_trigger
```

Dropping a trigger

- Here is how a database trigger is dropped:
- **Syntax:**
- drop trigger<trigger_name>
- **Example:**
- D
- drop trigger sales1_trigger

Types of Relational operation



1. Select Operation:

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

1. Notation: σ p(r)

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

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

- σ BRANCH_NAME="perryride" (LOAN)
- **Output:**

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

2. Project Operation:

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

1. Notation: ∏ A1, A2, An (r)

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

Example: CUSTOMER RELATION

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

- **Input:**

1. ∏ NAME, CITY (CUSTOMER)

- **Output:**

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

3. Union Operation:

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

1. Notation: R ∪ S

- A union operation must hold the following condition:
- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Example:
DEPOSITOR RELATION

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

BORROW RELATION

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

- **Input:**

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

- **Output:**

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

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

1. Notation: $R \cap S$

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

- **Input:**

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

- **Output:**

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

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

1. Notation: R - S

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

- **Input:**

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

- **Output:**

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product

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

1. Notation: E X D

Example:

EMPLOYEE

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

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

- **Input:**

1. EMPLOYEE X DEPARTMENT

- **Output:**

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

7. Rename Operation:

- The rename operation is used to rename the output relation.
- It is denoted by **rho** (ρ).
- **Example:** We can use the rename operator to rename STUDENT relation to STUDENT1.

1. ρ(STUDENT1, STUDENT)

What is RDBMS

- **RDBMS** stands for *Relational Database Management Systems*..
- All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.
- It is called Relational Data Base Management System (RDBMS) because it is based on relational model introduced by E.F. Codd.

How it works

- Data is represented in terms of tuples (rows) in RDBMS.
- Relational database is most commonly used database. It contains number of tables and each table has its own primary key.
- Due to a collection of organized set of tables, data can be accessed easily in RDBMS.

Brief History of RDBMS

- During 1970 to 1972, E.F. Codd published a paper to propose the use of relational database model.
- RDBMS is originally based on that E.F. Codd's relational model invention

What is table

- The RDBMS database uses tables to store data.
- A table is a collection of related data entries and contains rows and columns to store data.
- A table is the simplest example of data storage in RDBMS.
- Let's see the example of student table.

Example

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech
2	aryan	20	C.A
3	Mahesh	21	BCA
4	Ratan	22	MCA
5	Vimal	26	BSC

What is field

- Field is a smaller entity of the table which contains specific information about every record in the table.
- In the above example, the field in the student table consist of id, name, age, course.

What is row or record

- A row of a table is also called record. It contains the specific information of each individual entry in the table.
- It is a horizontal entity in the table.
- For example: The above table contains 5 records.

Let's see one record/row in the table.

1	Ajeet	24	B.Tech

What is column

- A column is a vertical entity in the table which contains all information associated with a specific field in a table.
- For example: "name" is a column in the above table which contains all information about student's name.

Ajeet
Aryan
Mahesh
Ratan
Vimal

NULL Values

- The NULL value of the table specifies that the field has been left blank during record creation.
- It is totally different from the value filled with zero or a field that contains space.

Data Integrity

- There are the following categories of data integrity exist with each RDBMS:
- **Entity integrity**: It specifies that there should be no duplicate rows in a table.
- **Domain integrity**: It enforces valid entries for a given column by restricting the type, the format, or the range of values.
- **Referential integrity**: It specifies that rows cannot be deleted, which are used by other records.
- **User-defined integrity**: It enforces some specific business rules that are defined by users. These rules are different from entity, domain or referential integrity.