DBMS

UNIT-2 Relational Data Model

DBMS

- ❖ DBMS stores data as file
- ❖ Data elements need to access individually.
- ❖ No relationship between data
- ❖ Normalization is not present
- ❖ DBMS does not support distributed database.
- ❖ It deals with small quantity of data.
- ❖ Data redundancy is common in this model.
- ❖ It supports single user.
- ❖ Low software and hardware necessities.
- ❖ Eg:-Examples: XML, Window Registry, etc.

RDBMS

- ❖ RDBMS stores data in tabular form.
- ❖ Multiple data elements can be accessed at the same time.
- ❖ Data is stored in the form of tables which are related to each other
- ❖ Normalization is present.
- ❖ RDBMS supports distributed database.
- ❖ It deals with large amount of data.
- ❖ Keys and indexes do not allow Data redundancy.
- ❖ It supports multiple users.
- ❖ Higher software and hardware necessities.
- ❖ Eg;-Examples: MySQL, PostgreSQL, SQL Server, Oracle, Microsoft Access etc.

2.1 Fundamentals of RDBMS

1. Introduction

The **Relational model** stores data in the form of tables. This concept is introduced by Dr. E.F. Codd, a researcher of IBM. The relational model is the first choice of commercial data processing applications for storing the data.

2.Definition

A **relational database** is collectively combination of data structures, storage and retrieval operations and integrity constraints.

3.Advantages of Relational Model

1. Ease of use
2. Flexibility
3. Security
4. Data Independence
5. Data Manipulation Language

2.1 Fundamentals of RDBMS

4.Characteristics of Relational Database

1. The data is stored in the tables which are having relationships in between them.
2. systematic arrangement of data into rows and columns, called as relation or table.
3. A table is also form in two-dimensional structure.
4. every cell in the relation there is one and only one value which is known as scalar value.
5. Column represents attribute, and each column has a distinct name.
6. All values entered in the columns are of the same data format.
7. It supports operation like data definition, manipulation and transaction management.

Basic concept of RDBMS

- **Files**-collection of Related Record.
- **Tables**-collection of related data entries and it consist of Column and rows.
- **Fields**-smallest unit of data and has a meaning to the user eg name ,address
- **Data types**- data type define what kind of value a column can hold such as integer ,character etc.
- **Tuples** -a row of table is also called as record or tuple.
- **Null value** - missing or undefined value

Edgar.F Codd Rule for RDBMS

- Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.
- These rules can be applied on any database system that manages stored data using only its relational capabilities

Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some **table cell.** Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value).

Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.Primary key must not be null.For Eg:- Landline No.

Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation. **Eg :- SQL**

Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications. For Eg :- Location change,Rename File.

Rule 9: Logical Data Independence

The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. **For example,** if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

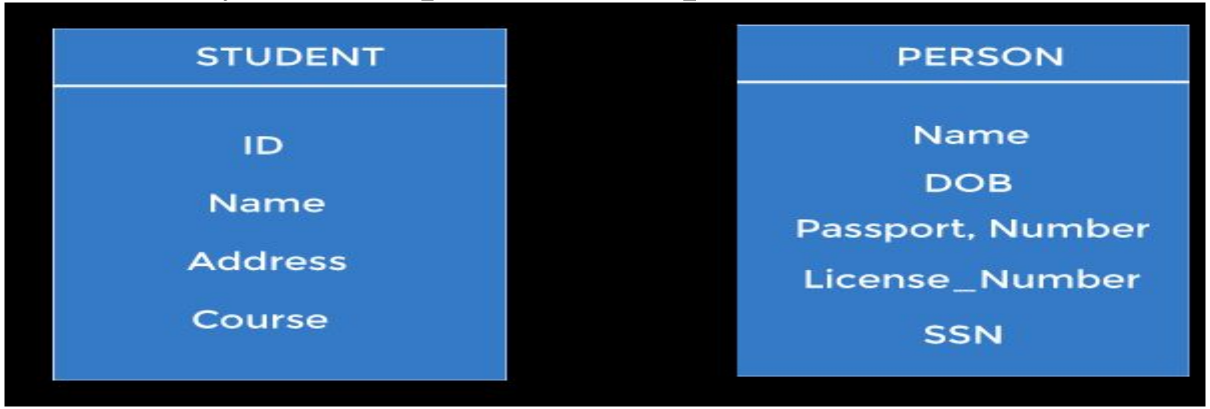
Rule 12: Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

For Eg. High level language to machine language.

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,** ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person



Types of Keys in DBMS (Database Management System)

There are mainly Eight different types of Keys in DBMS and each key has it's different functionality:

- ❖ Super Key
- ❖ Primary Key
- ❖ Candidate Key
- ❖ Foreign Key
- ❖ Composite Key

Super Keys

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.

For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME),

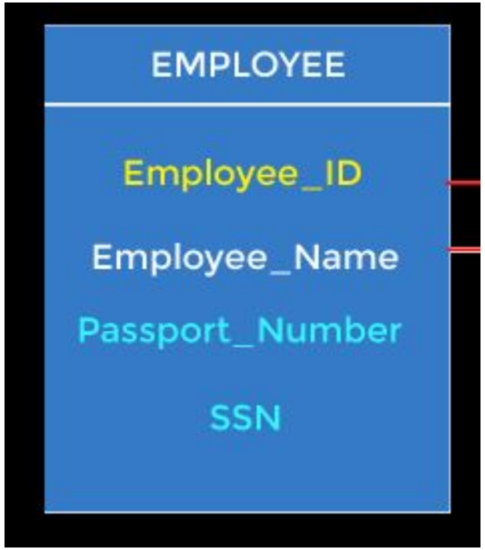
the name of two employees can be the same,

but their EMPLYEE_ID can't be the same.

Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID

(EMPLOYEE_ID, EMPLOYEE-NAME), etc.



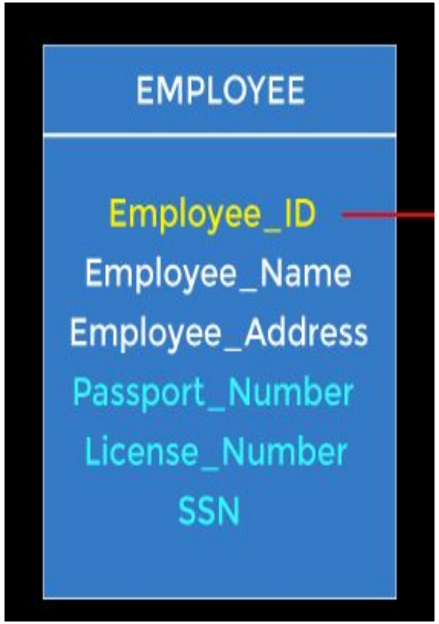
Primary Keys

PRIMARY KEY in dbms is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. A table cannot have more than one primary key.

Rules for defining Primary key:

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified

  or updated if any foreign key refers to that primary key.



CANDIDATE KEY in SQL is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes. The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key. A table can have multiple candidate keys but only a single primary key.

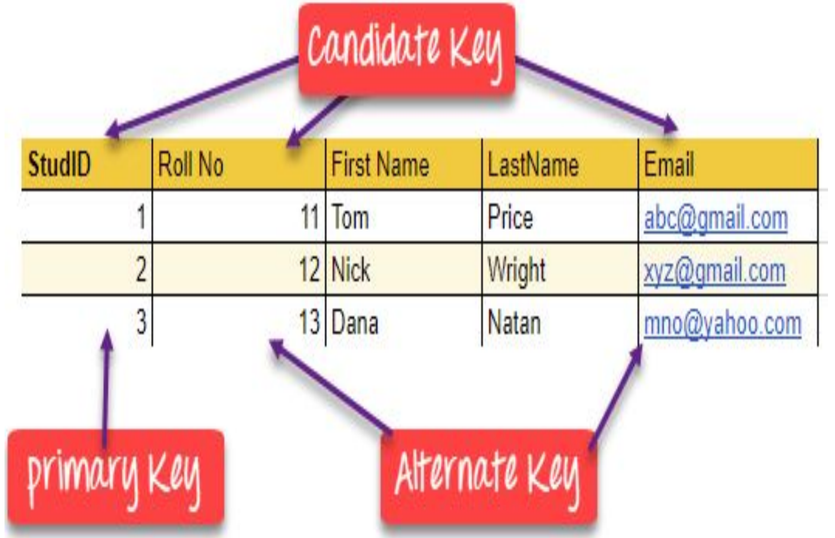
Properties of Candidate key:

- It must contain unique values
- Candidate key in SQL may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table

Candidate key Example: In the given table Stud ID,

Roll No, and email are candidate keys which help

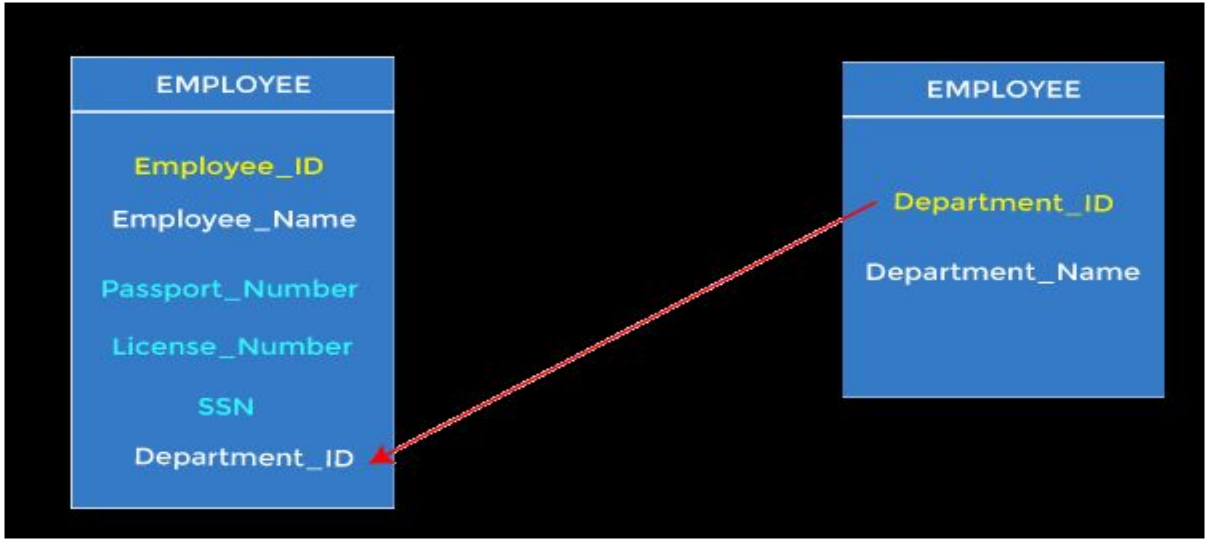
us to uniquely identify the student record in the table.



StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

FOREIGN KEY is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table.

This concept is also known as Referential Integrity



Composite Key

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

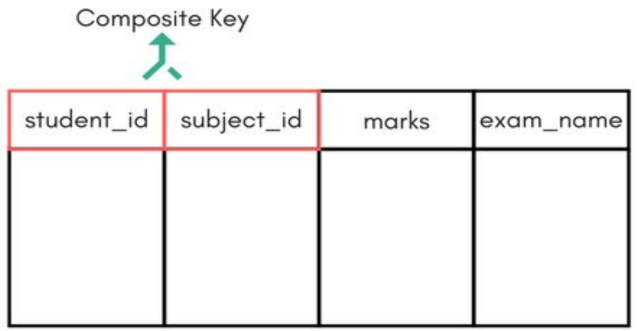
In the above picture we have a **Score** table which stores the marks scored by a student in a particular subject.

In this table student_id and subject_id

together

will form the primary key,

hence it is a composite key.



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

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.
- 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 are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

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.

Normalization Advantages

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Increased storage efficiency
- A better handle on database security
- Avoid data modification (Insert/delete/Update) anomalies

Normalization Disadvantages

- 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
- Maintenance overhead.
- Requires more joins to get the desired result.

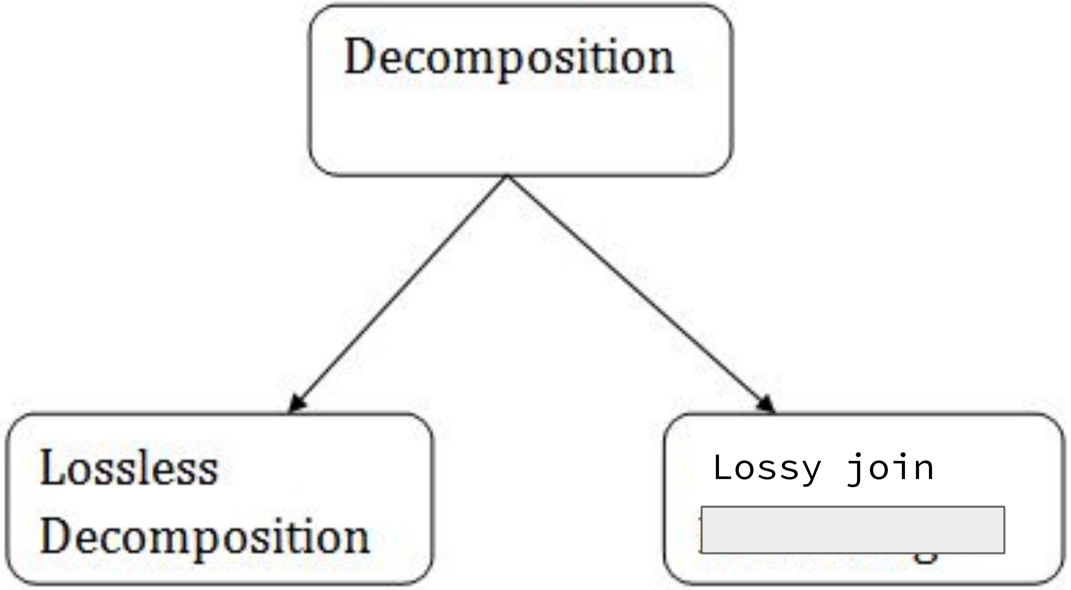
Normalization concepts

1. Decomposition
2. Dependency Preservation
3. Functional Dependency
4. Multivalued dependency

Decomposition

- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

Decomposition



Lossless Decomposition

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.
- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.
- R=R1 U R2 U R3

Lossless Decomposition

Example:

Model No	Price	Make
1	110	canon
2	200	sony
3	300	canon

Model No	Make
1	Canon
2	sony
3	canon

Price	Make
110	canon
200	sony
300	canon

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.

Multivalued Dependency

Bike Model	Manu_year	Colour
M201	2008	WHITE
M201	2008	BLACK
M401	2013	BLACK
M401	2013	WHITE
M501	2017	WHITE

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 → → MANUF_YEAR
2. BIKE_MODEL → → COLOR

Functional Dependency

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

$A \rightarrow B$ Eg PTS

$B \rightarrow C$

$C \rightarrow A$

Anormalies

- Row level
- Column Level

SID	Sname	CID	CName	FID	FNAME	Salary
1	Ram	C1	dbms	F1	John	30000
2	Ravi	C2	DTE	F2	Bob	40000
3	Ramesh	C3	OOP	F3	Hary	35000
4	Rahul	C1	dbms	F1	John	30000
5	Raj	C2	DTE	F2	Bob	40000

Need of Normalization

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

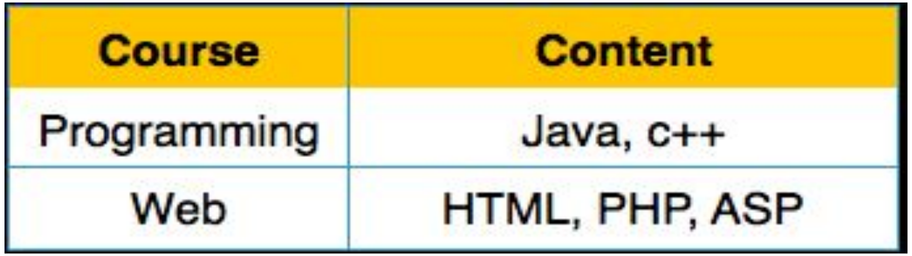
- **Update anomalies** − If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.**Eg- Martial Status**
- **Deletion anomalies** − We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.Eg:-Mobile No.
- **Insert anomalies** − We tried to insert data in a record that does not exist at all.

Normal Forms

- 1NF
- 2NF
- 3NF
- BCNF (Boyce Codd's normal form)
- 4NF
- 5NF

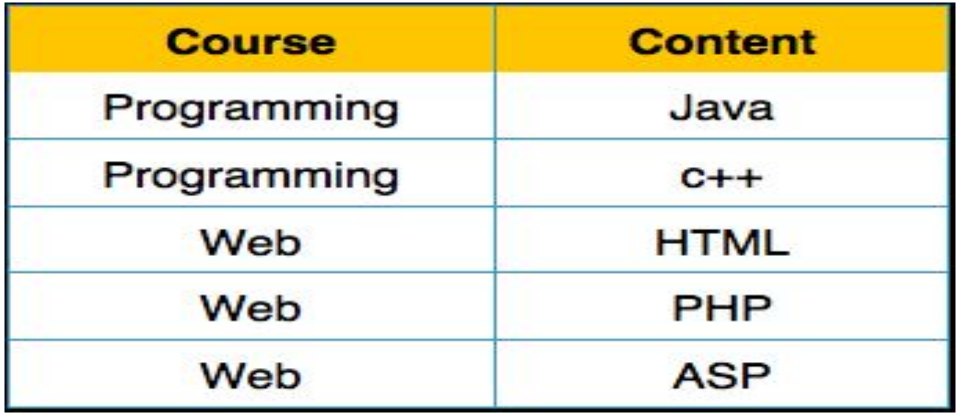
1NF First Normal Forms

- Each table cell should contain a single value.
- Each record needs to be unique.



Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

We re-arrange the relation (table) as below, to convert it to First Normal Form.



Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

2NF Second Normal Forms

- 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

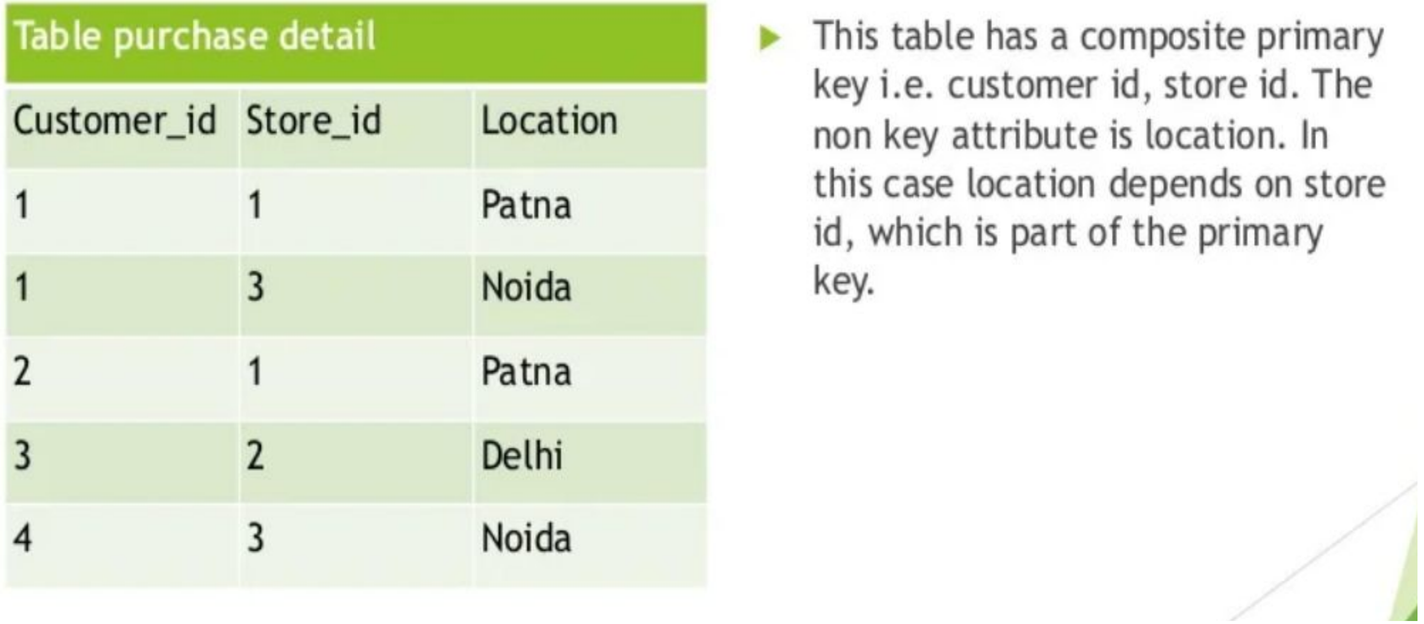


Table purchase detail

Customer_id	Store_id	Location
1	1	Patna
1	3	Noida
2	1	Patna
3	2	Delhi
4	3	Noida

- This table has a composite primary key i.e. customer id, store id. The non key attribute is location. In this case location depends on store id, which is part of the primary key.

After decomposing it into second normal form it looks like:

Table Purchase	
Customer_id	Store_id
1	1
1	3
2	1
3	2
4	3

Table Store	
Store_id	Location
1	Patna
2	Delhi
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 $X \rightarrow Y$.

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

THIRD NORMAL FORM

Table Book Details

Bood_id	Genre_id	Genre type	Price
1	1	Fiction	100
2	2	Sports	110
3	1	Fiction	120
4	3	Travel	130
5	2	sports	140

- In the table, book_id determines genre_id and genre_id determines genre type. Therefore book_idd determines genre type via genre_id and we have transitive functional dependency.

After decomposing it into third normal form it looks like:

TABLE BOOK		
Book_id	Genre_id	Price
1	1	100
2	2	110
3	1	120
4	3	130
5	2	140

TABLE GENRE	
Genre_id	Genre type
1	Fiction
2	Sports
3	Travel

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.
- SQL is also pronounced as Sequel

SQL

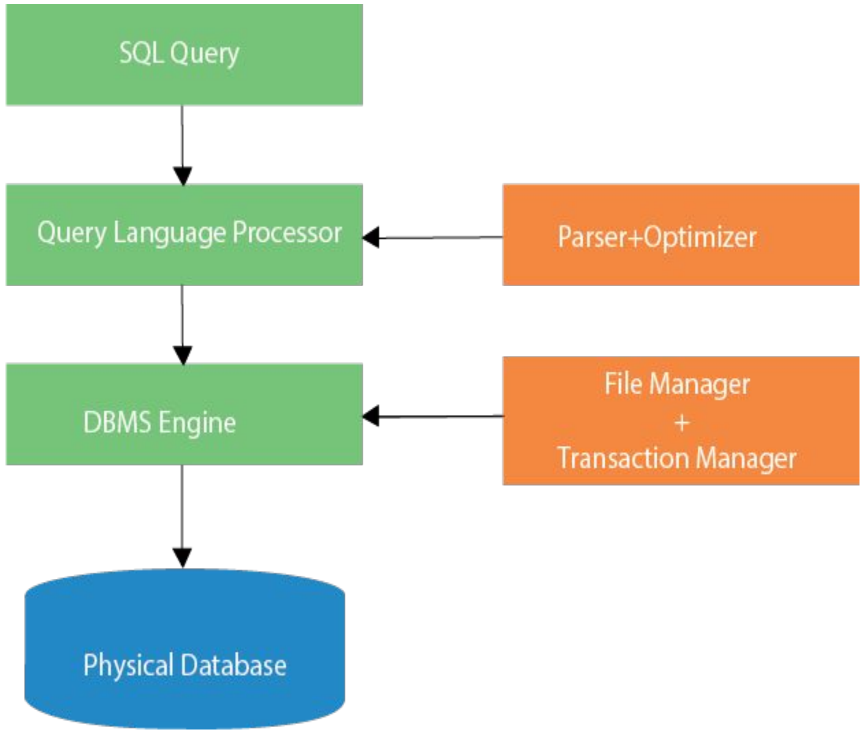
Rules:

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 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.

1)Char(n): Fixed-length character data (string),n characters long.Maximum size-255

2)varchar2(n):variable length character string,Maximum size-2000

3)date:date date type for storing date and time.

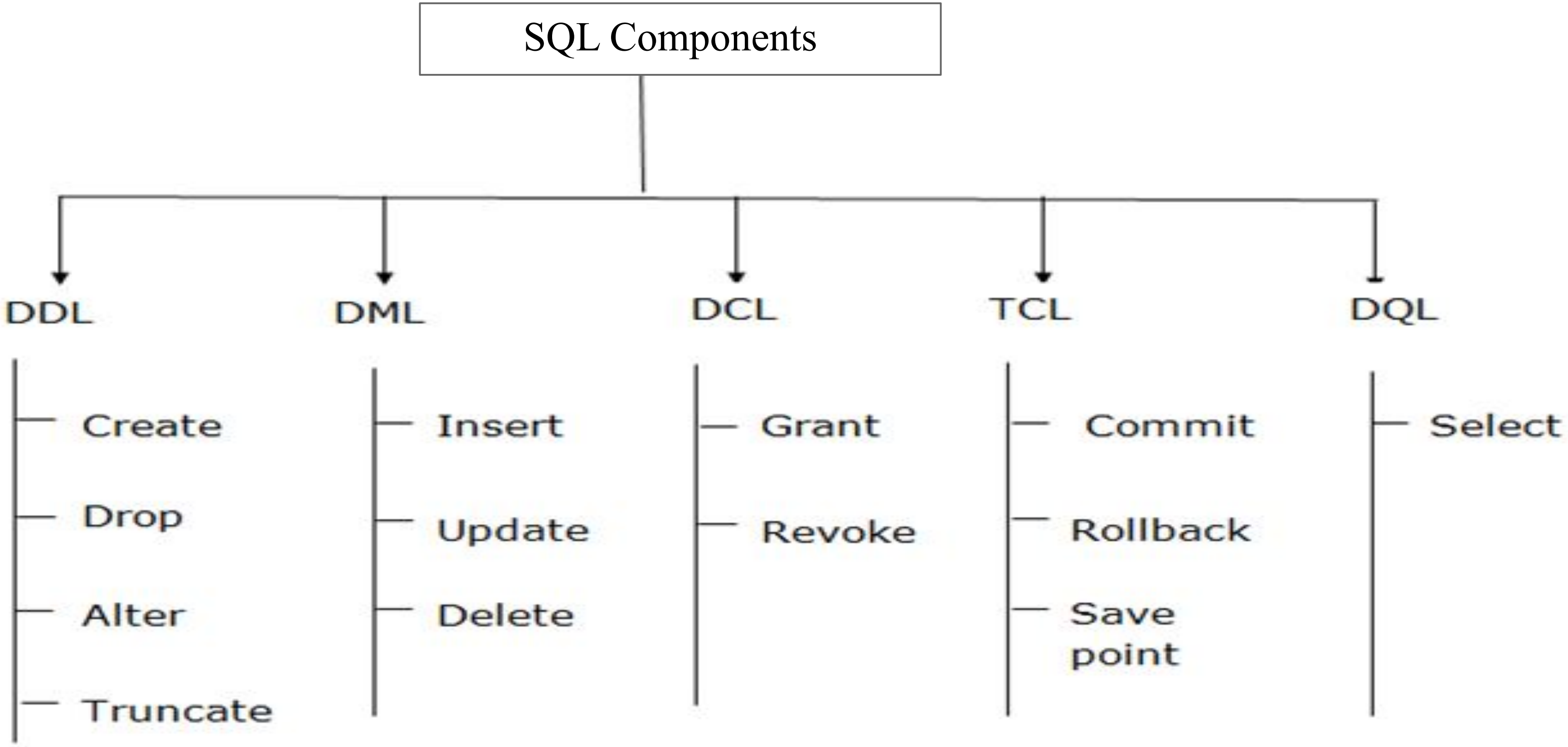
The default format for the date is dd-mm-yy

4)long:-character data up to a length of 2 gb.only one long column is allowed per table

5)number(o,d):-number data type for integer and real o=overall no of digits , d=number of digits to the right of the decimal point max value o=38, d=-84 to +127

Eg number (5,2)=999.99

SQL Components



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
- Rename
- DESC

a. CREATE It is used to create a new table in the database.

CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

Ex:-CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

b.**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.

To add a new column in the table

ALTER TABLE table_name ADD column_name COLUMN-definition;

To modify existing column in the table:

ALTER TABLE table_name MODIFY(column_definitions....);

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

c.DROP: It is used to delete both the structure and record stored in the table.

```
DROP TABLE table_name;
```

```
DROP TABLE EMPLOYEE;
```

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

```
TRUNCATE TABLE table_name;
```

```
TRUNCATE TABLE EMPLOYEE;
```

e. Rename:

Rename table command is used to rename the old table name to new table name.

Syntax:- rename table old_tablename to new_tablename;

```
Rename emp to emp1;
```

F. Desc

The table structure can be described by using describe command.

Alter

Drop column

Alter table table_name drop column column_name;

Ex :- Alter table student drop (Age));

Modify

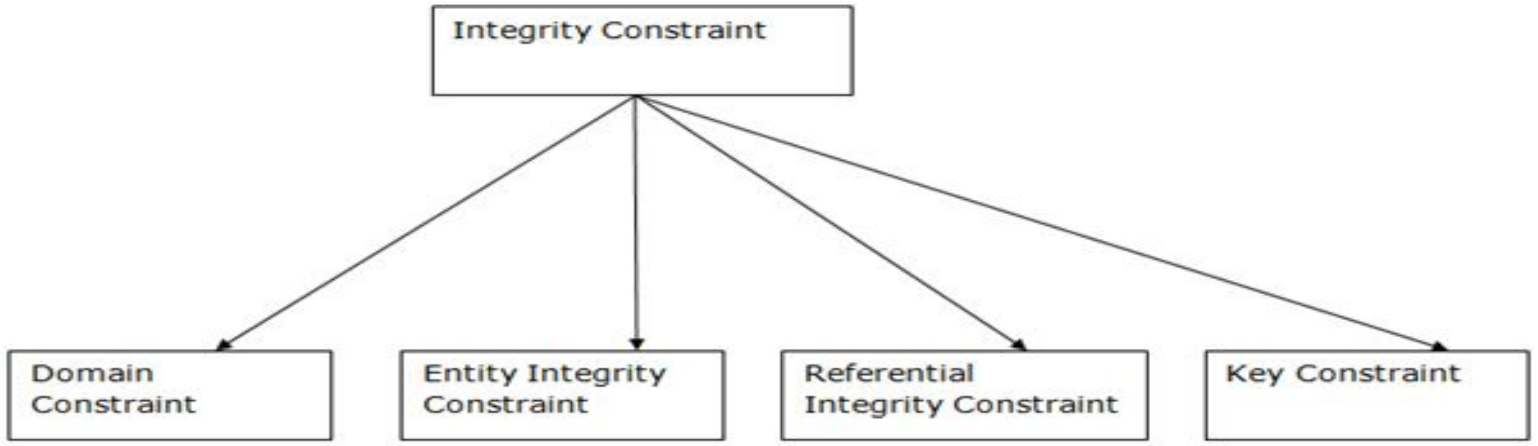
Alter table table_name rename column column_name TO new column_name;

ALTER table dept rename column dept_name TO name;

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.

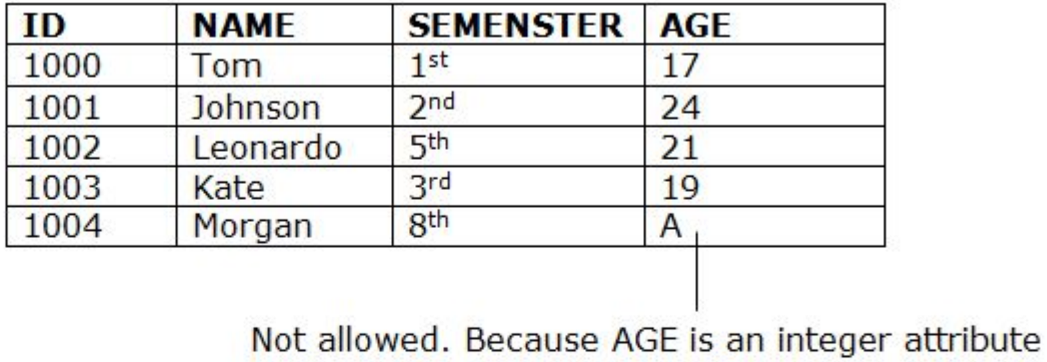
Types of Integrity Constraint



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:

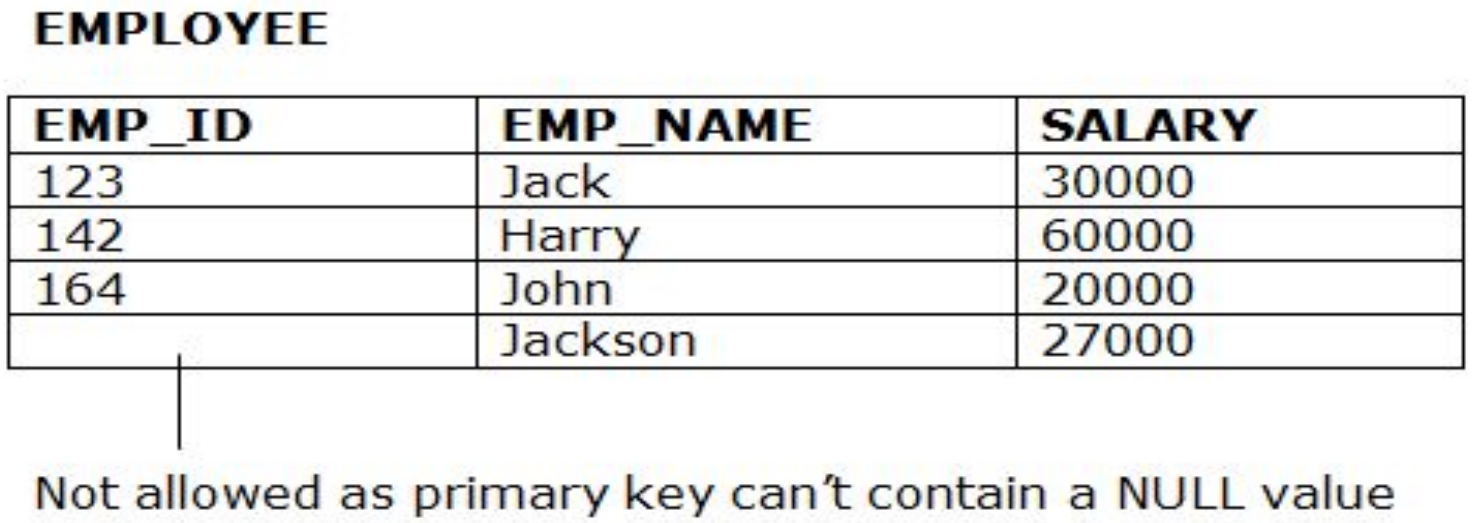


ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

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.



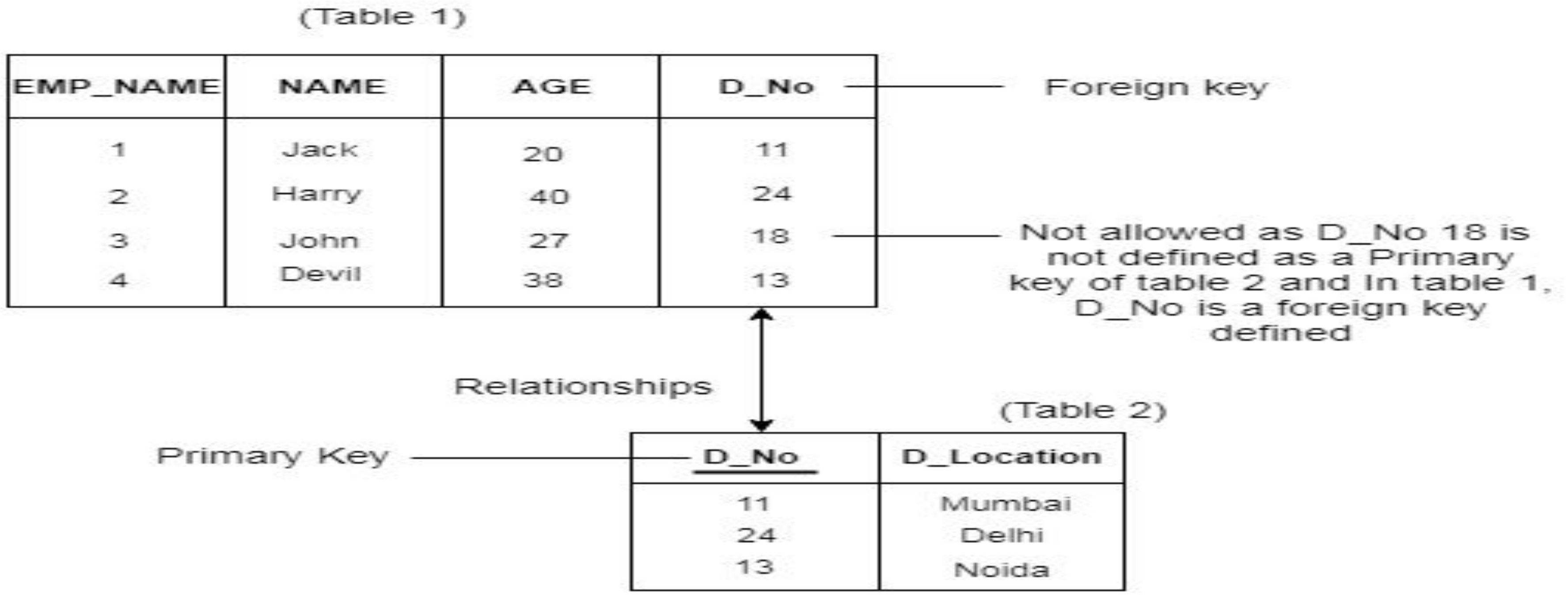
EMPLOYEE

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.



(Table 1)

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

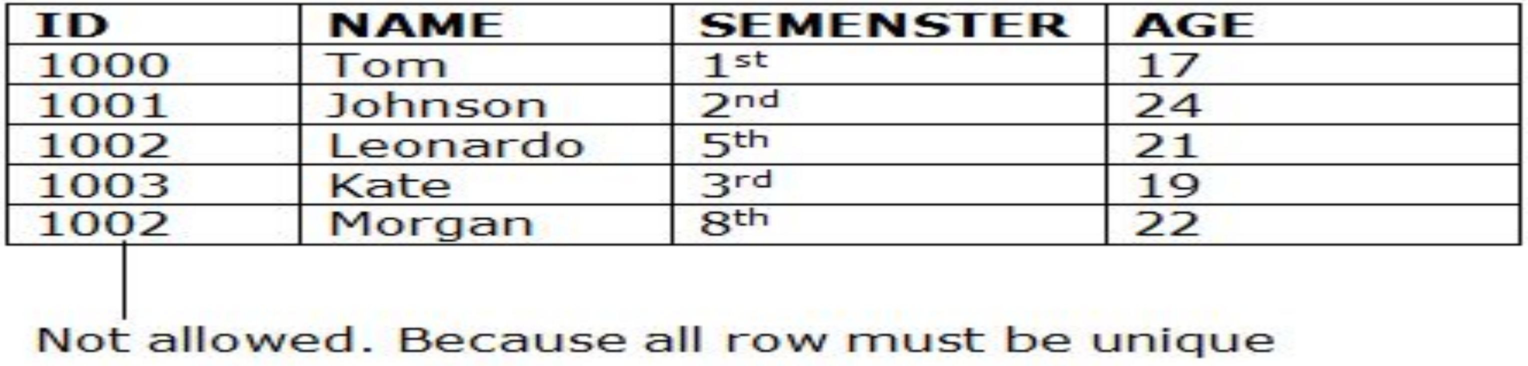
Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined

(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.



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

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
- SELECT

a. INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:-INSERT INTO TABLE_NAME   VALUES (value1, value2, value3, ... valueN);

INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");

INSERT INTO javatpoint values (“Sonoo”,”DBMS”);

b. UPDATE: This command is used to update or modify the value of a column in the table.

UPDATE table_name SET column_name=new_value WHERE column_name=some_value;

Syntax:- UPDATE students   SET User_Name = 'Sonoo'   WHERE Student_Id = '3'

c. DELETE: It is used to remove one or more row from a table.

DELETE FROM table_name [WHERE condition];

DELETE FROM javatpoint WHERE Author="Sonoo";

Delete * from table_name; (AR)

3. Data Control Language

DCL commands are used to grant and take back authority from any database user. DCL commands are related to the security of the database.

Here are some commands that come under DCL:

- Grant
- Revoke

a. Grant: The objects created by one user are not accessible by another user unless the owner of those object gives such permission to the other user.It is used to give user access privileges to a database. List of object are ALTER, DELETE, INSERT, SELECT,UPDATE with GRANT OPTION

Syntax: Grant {object privileges} ON object name TO user name [with GRANT OPTION]

Syntax:-GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER;

GRANT ALL ON student TO Sachin;

3. Data Control Language

b. Revoke: The revoke statement is used to deny the grant given on an object. It is used to take back permissions from the user.

Syntax:-
```
REVOKE {object privileges}
    ON object name
    FROM user_name;
```

Syntax:-REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

REVOKE DELETE ON student From sachin;

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:

SQL>DELETE FROM CUSTOMERS WHERE AGE = 25;

SQL>COMMIT;

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

DELETE FROM CUSTOMERS WHERE AGE = 25;

ROLLBACK;

c. SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

SAVEPOINT SAVEPOINT_NAME;

Example : savepoint sp1;

SQL>>Delete from customers where ID=3;

SQL>>Rollback sp2

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: SELECT column_name FROM TABLES   WHERE conditions;

Eg:-Select lastname from person where city=”pune”;

Select * from person;

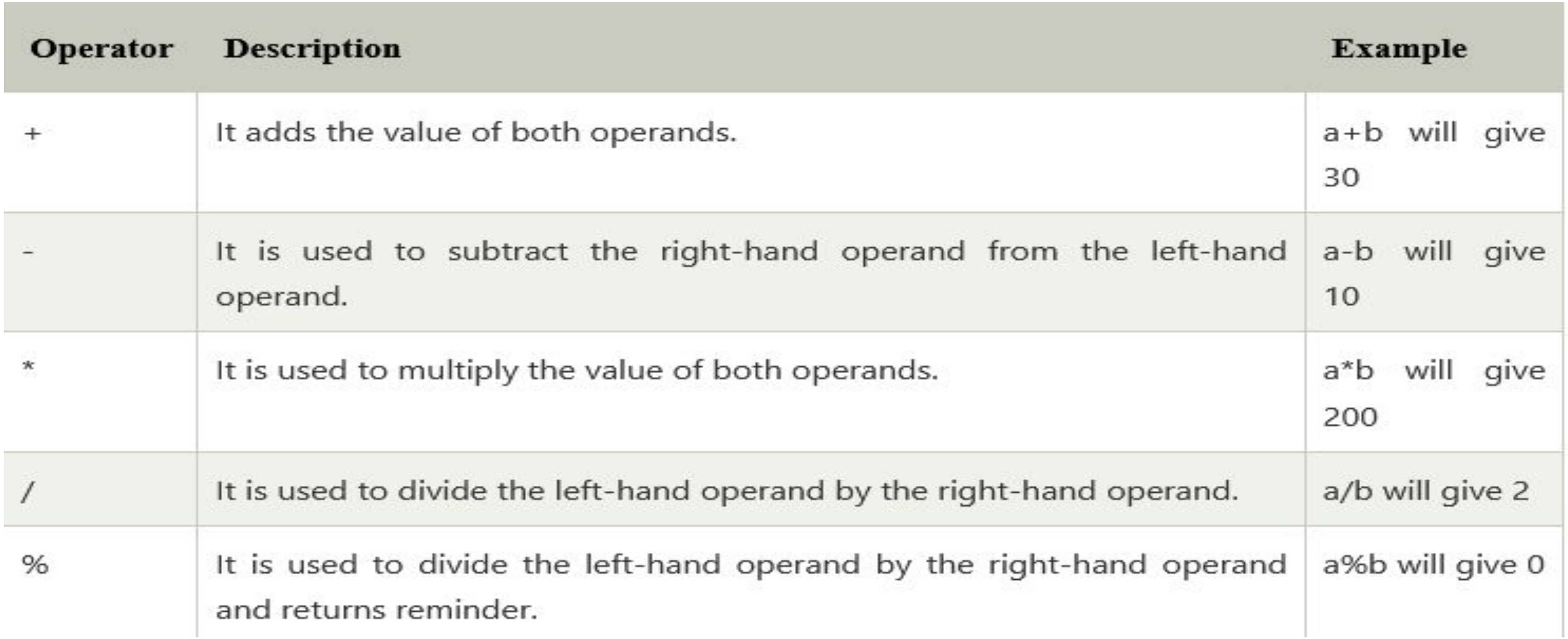
What is an Operator in SQL?

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

- Arithmetic operators
- Comparison operators
- Logical operators
- Range Searching Operators
- Pattern Matching operator
- Set Operators

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 Arithmetic Operators

Examples :

Display salary of employees by adding 5000 to it.

```
Sql>> select salary  + 5000 from emp;
```

Display salary of employee by subtracting 5000 to it. **Answer**

Display salary of employee by giving 5% raise in salary to employee.

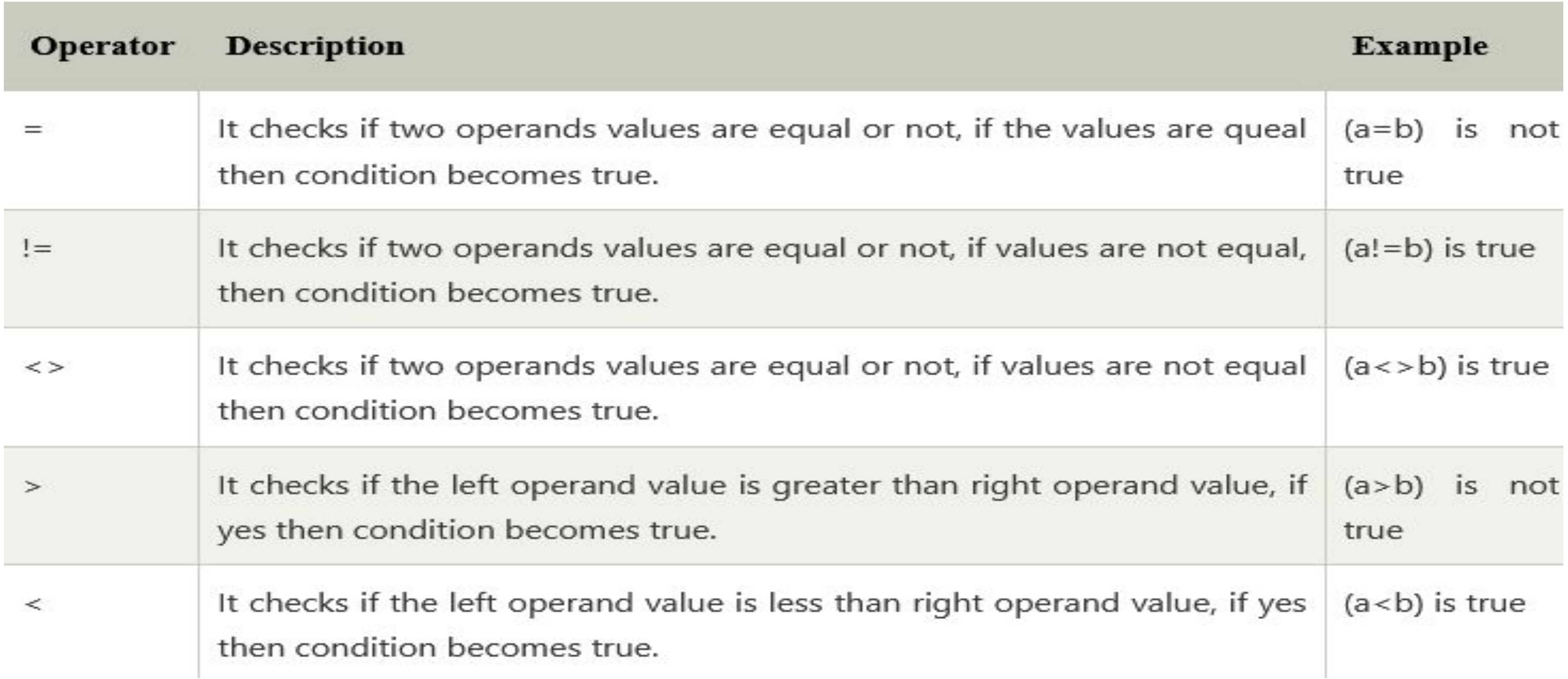
```
>>Select salary + 0.05 * salary from employee;
```

Display salary by dividing the salary by 2,

```
>>Select salary/2 from employe;
```

SQL Comparison Operators:

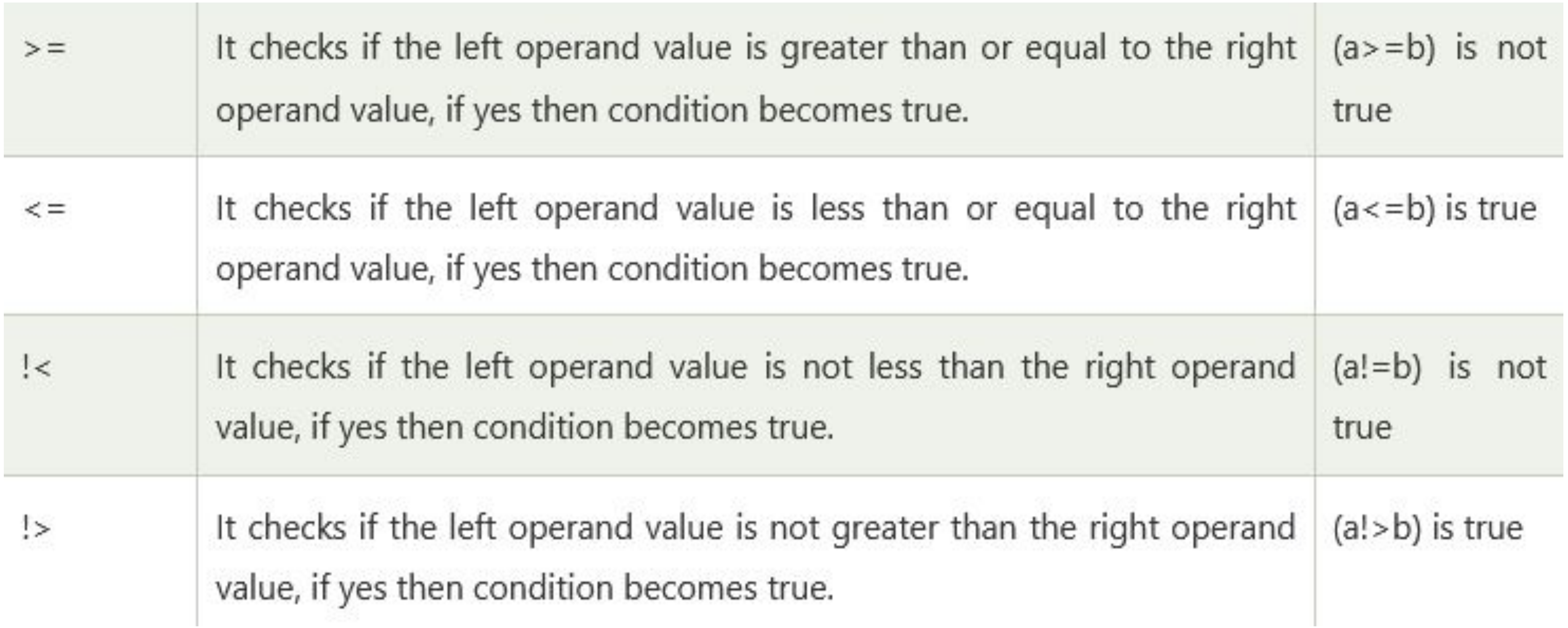
Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.



Operator	Description	Example
=	It checks if two operands values are equal or not, if the values are queal 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

SQL Comparison Operators:

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



>=	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 true.	(a!>b) is true

SQL Comparison Operators:

Company	Order_Number
sega	3412
wipro	2312
TCS	4678
wipro	6798

Eg. Find company of the order_Number 6789
>> select company from order where order_number=6789;

Find company for order_number between 4000 to 7000.
>> Select company from orders where order_number >=4000 and ordernumber>=7000;

SQL Logical Operators

AND and OR join two or more conditions in a WHERE clause.The AND operator display a row if all conditions listed are true.The OR operator displays a row if ANY of the conditions listed are true.

The AND operator allows the existence of multiple conditions in any sql statements WHERE clause.

LastName	**FirstName**	**Address**	**City**
sharma	Arun	Camp 23	Mumbai
Mane	Vijay	GM Road18	Mumbai
Mane	Pooja	Decaan 19	Mumbai

Example1: Find a person with firstname equal to “Pooja **and** the lastname equal to “mane”;
sql>> Select * from persons where firstname=”Pooja” and lastname=”mane”

Example1: Find a person with firstname equal to “Pooja **or** the lastname equal to “mane”;
sql>> Select * from persons where firstname=”Pooja” or lastname=”mane”

Combine And OR :- Select * from persons WHERE(Firstname=”Pooja or Firstname=”Vijay” AND Lastname=”Mane”);

Range Searching Operators

IN Operator

The IN operator may be used if you know the exact value you want to return for at least one of the Columns

Syntax:-select column_name from table_name WHERE column_name IN(value1,value2.);

Example :- To display the persons name with LastName equal to “Sharma” or “Johnson” use the following SQL

Syntax:- select * from Persons WHERE LastName IN(‘Sharma’,’Johnson’);

LastName	**FirstName**	**Address**	**City**
sharma	Arun	Camp 23	Mumbai
Mane	Vijay	GM Road18	Mumbai
Mane	Pooja	Decaan 19	Mumbai

Range Searching Operators

Not IN Operator:- Not IN is negative operator

Syntax:-select column_name from table_name WHERE column_name NOT IN(value1,value2.);

Example :- To display the persons name with LastName NOT equal to “Sharma” or “Johnson” use the following SQL

Syntax:- select * from Persons WHERE LastName NOT IN(‘Sharma’,’Johnson’);

LastName	**FirstName**	**Address**	**City**
sharma	Arun	Camp 23	Mumbai
Gupta	Vijay	GM Road18	Mumbai
Mane	Pooja	Decaan 19	Mumbai

Range Searching Operators

Between and Not Between Operator:-The between and not between operator select a range of data between two values and not between the two values.These values can be number, text or dates.

The between operator is used to search for value that are within that a set of values, given the minimum value and the maximum value.

Syntax:-select column_name from table_name WHERE column_name between value1 and value2;

Syntax:-select column_name from table_name WHERE column_name not between value1 and value2;

Range Searching Operators

LastName	**FirstName**	**Address**	**City**
sharma	Arun	Camp 23	Mumbai
Gupta	Vijay	GM Road18	Pune
Mane	Pooja	Decaan 19	Mumbai

Example :- To display the persons alphabetically **between** (and including) “sharma” and exclusive “johnson” use the following sql

Select * from persons where LastName Between ‘ Sharma’ and ‘Mane’;

Example :- To display the persons outside the range used the not operator:

Select * from persons where LastName Not Between ‘ Sharma’ and ‘Mane’;

Pattern Matching in SQL

- LIKE clause is used to perform the pattern matching task in SQL.
- A WHERE clause is generally preceded by a LIKE clause in an SQL query.
- LIKE clause searches for a match between the patterns in a query with the pattern in the values present in an SQL table. If the match is successful, then that particular value will be retrieved from the SQL table.
- LIKE clause can work with strings and numbers.

The LIKE clause uses the following symbols known as wildcard operators in SQL to perform this pattern-matching task in SQL.

1. To represent zero, one or more than one character, % (percentage) is used.
2. To represent a single character _ (underscore) is used

ID	Name	City	Salary	Age
1	Priyanka Bagul	Nasik	26000	20
2	Riya Sharma	Mumbai	72000	28
3	Neha Verma	Varanasi	37000	19
4	Neeta Desai	Nasik	39500	21
5	Priya Wagh	Udaipur	60000	32

A) Using LIKE clause with % (percentage) (Draw Table)

Example 1: Write a query to display employee details in which name starts with 'Pr

SELECT * **FROM** employee_details **WHERE Name** LIKE 'Pr%'; '

Example 2: Write a query to display employee details in which 'ya' is a substring in a name.

SELECT * **FROM** employee_details **WHERE Name** LIKE '%ya%';

Write a query to display employee details in which city name ends with 'i'.;

```
SELECT * FROM employee_details WHERE City LIKE '%i';
```

Example 4: Write a query to display employee details in which age number starts with 2.

```
SELECT * FROM employee_details WHERE Age LIKE '2%';
```

Example 5:

Write a query to display employee details in which salary contains a number 50 in between.

```
SELECT * FROM employee_details WHERE Salary LIKE '%50%';
```

(B) Using LIKE clause with _ (underscore)

Example 1:

Write a query to display employee details in which city name starts with 'Na', ends with 'ik', and contains any single character between 'Na' and 'ik'.

Query:

```
mysql> SELECT * FROM employee_details WHERE City LIKE 'Na_ik';
```

Example 2:

Write a query to display employee details in which salary contains a number starting with '3' succeeding any two digits and finally ends with '00'.

```
SELECT * FROM employee_details WHERE Salary LIKE '3__00';
```

(C) Using LIKE clause with % and _ operator in a single query

Example 1: Write a query to display employee details in which employee name contains 'a' at fifth position.

Query:

```
1.  mysql> SELECT * FROM employee_details WHERE Name LIKE '____a%';
```

Example 2:

Write a query to display employee details in which salary contains a substring '00' starting from the 5th position.

Query:

```
1.  mysql> SELECT * FROM employee_details WHERE Salary LIKE '__00%';
```

(D) Using LIKE clause with NOT operator

Example 1:

Write a query to display employee details in which employee name is not like 'Priya'.

Query:

```
1. mysql> SELECT * FROM employee_details WHERE Name NOT LIKE 'Priya%';
```

Example 2:

Write a query to display employee details in which employee name is not like any single character followed by 'Na' and ending with 'ik'.

Query:

```
1. mysql> SELECT * FROM employee_details WHERE City NOT LIKE 'Na_ik';
```

SQL Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

Syntax

The basic syntax of a table alias is as follows.

```
SELECT column1, column2....FROM table_name AS alias_name WHERE [condition];
```

The basic syntax of a column alias is as follows.

```
SELECT column_name AS alias_name FROM table_name  WHERE [condition];
```

LastName	**FirstName**	**Address**	**City**
sharma	Arun	Camp 23	Mumbai
Gupta	Vijay	GM Road18	Pune
Mane	Pooja	Decaan 19	Mumbai

Example :Using a Column Alias:

Select LastName AS Family, Firstname AS Name from Persons:

Family	**Name**
sharma	Arun
Gupta	Vijay
Mane	Pooja

LastName	**FirstName**	**Address**	**City**
sharma	Arun	Camp 23	Mumbai
Gupta	Vijay	GM Road18	Pune
Mane	Pooja	Decaan 19	Mumbai

Example :Using a Table Alias:

Select LastName,Firstname from person AS Employee:

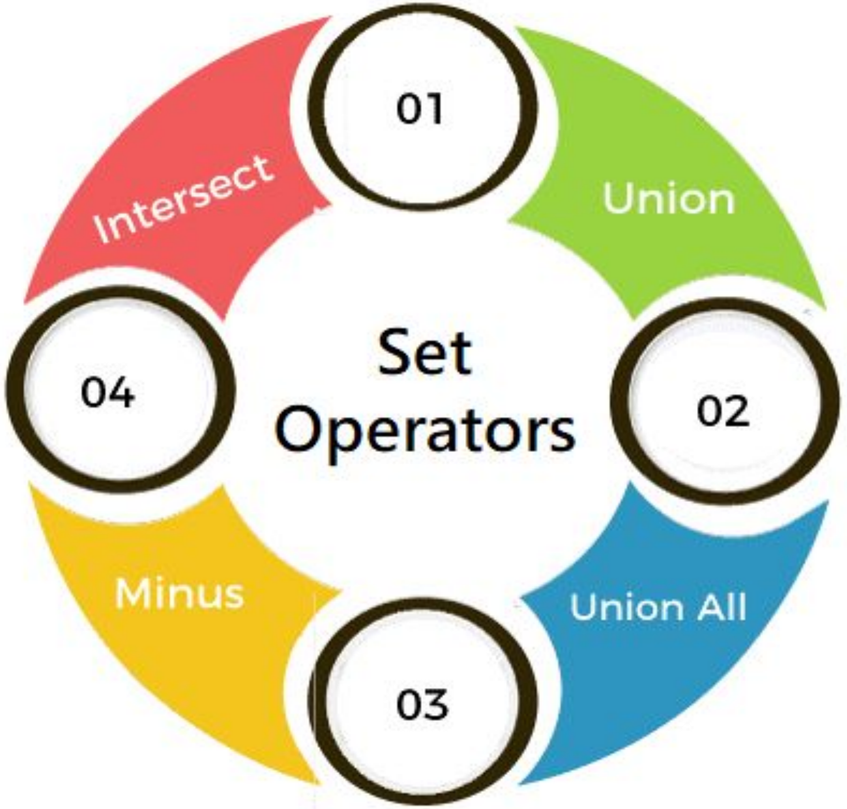
Employee

Lastname	**Firstname**
sharma	Arun
Gupta	Vijay
Mane	Pooja

SET Operators in SQL

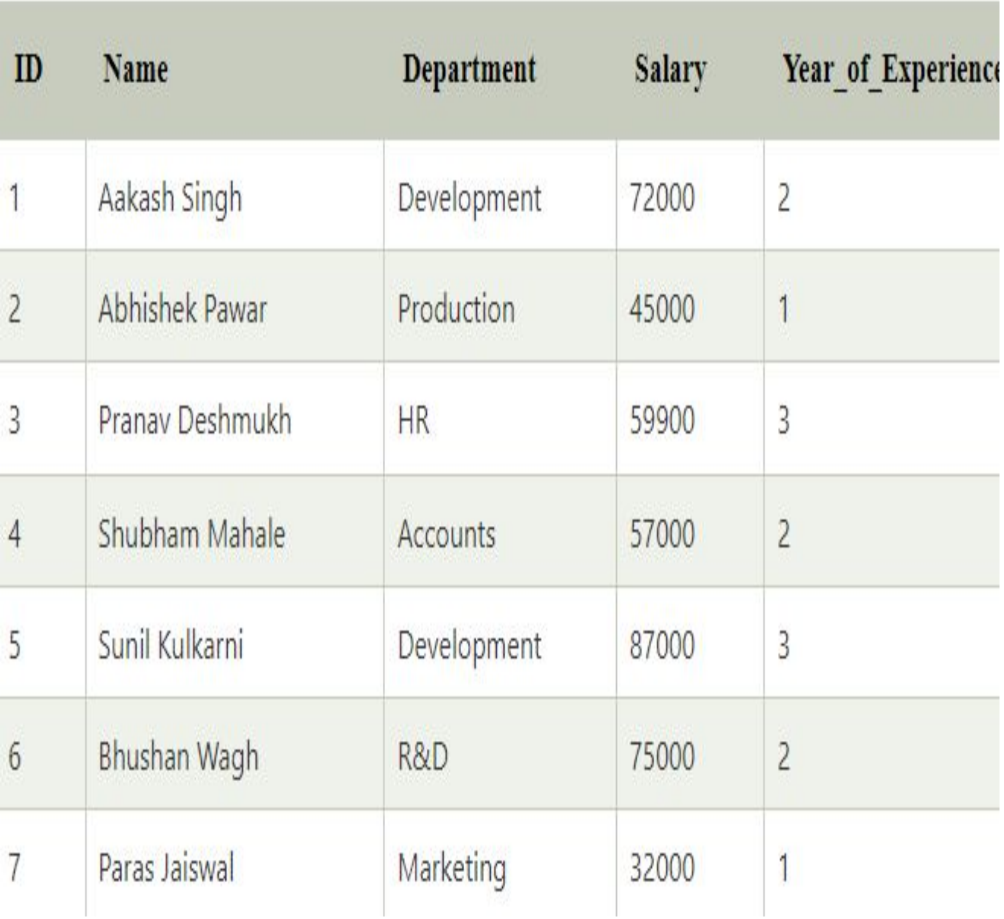
There are certain rules which must be followed to perform operations using SET operators in SQL. Rules are as follows:

1. **The number and order of columns must be the same.**
2. **Data types must be compatible.**

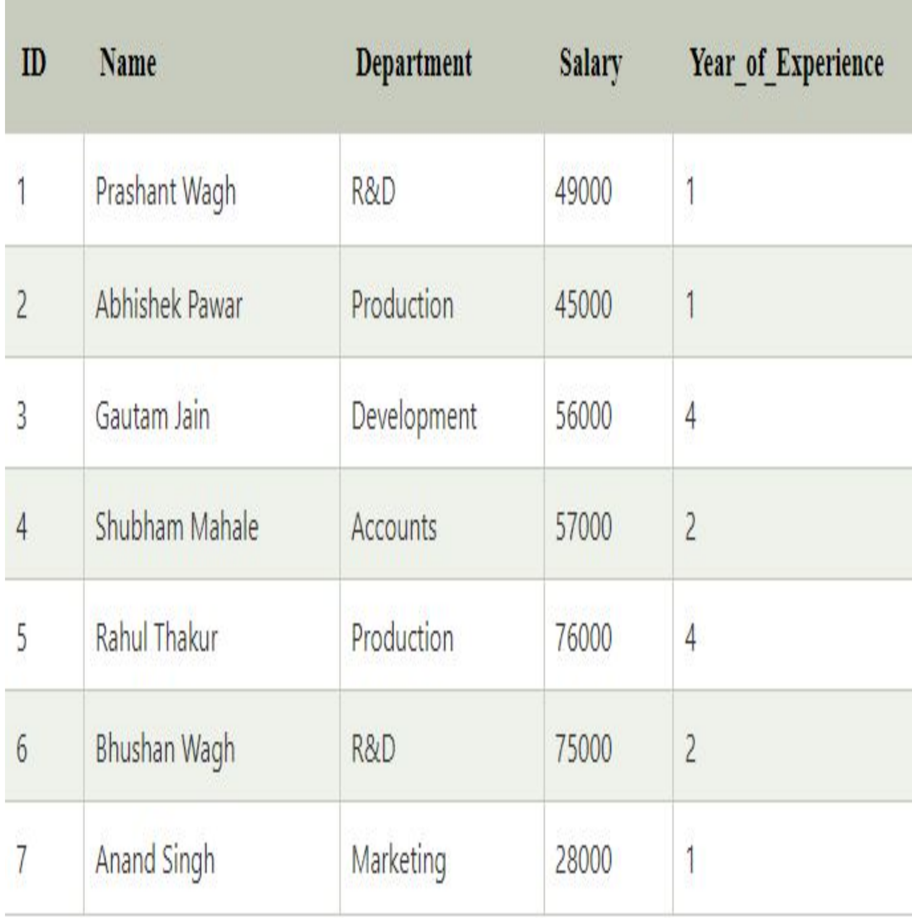


1. UNION:

- UNION will be used to combine the result of two select statements.
- Duplicate rows will be eliminated from the results obtained after performing the UNION operation.
- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order



ID	Name	Department	Salary	Year_of_Experience
1	Aakash Singh	Development	72000	2
2	Abhishek Pawar	Production	45000	1
3	Pranav Deshmukh	HR	59900	3
4	Shubham Mahale	Accounts	57000	2
5	Sunil Kulkarni	Development	87000	3
6	Bhushan Wagh	R&D	75000	2
7	Paras Jaiswal	Marketing	32000	1



ID	Name	Department	Salary	Year_of_Experience
1	Prashant Wagh	R&D	49000	1
2	Abhishek Pawar	Production	45000	1
3	Gautam Jain	Development	56000	4
4	Shubham Mahale	Accounts	57000	2
5	Rahul Thakur	Production	76000	4
6	Bhushan Wagh	R&D	75000	2
7	Anand Singh	Marketing	28000	1

SQL Statement>>SELECT * FROM t_employees UNION SELECT *FROM t2_employees;

2. UNION ALL:

- This operator combines all the records from both the queries.
- Duplicate rows will be not be eliminated from the results obtained after performing the UNION ALL operation.

Example 1:

Write a query to perform union all operation between the table t_employees and the table t2_employees.

Query:

1. mysql> SELECT *FROM t_employees UNION ALL SELECT *FROM t2_employees;

3. INTERSECT:

- It is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements.

Example 1:

Write a query to perform intersect operation between the table t_employees and the table t2_employees.

Query:

```
mysql> SELECT * FROM t_employees INTERSECT SELECT * FROM
t2_employees;
```

4. MINUS

- It displays the rows which are present in the first query but absent in the second query with no duplicates.

Example 1:

Write a query to perform a minus operation between the table t_employees and the table t2_employees.

Query:

1. mysql> SELECT * FROM t_employees MINUS SELECT * FROM t2_employees;

ID	Name	Department	Salary	Year_of_Experience
1	Aakash Singh	Development	72000	2
3	Pranav Deshmukh	HR	59900	3
5	Sunil Kulkarni	Development	87000	3
7	Paras Jaiswal	Marketing	32000	1

Since we have performed Minus operation between both the tables, so only the unmatched records from both the tables are displayed.