Unit-2

Syllabus:

Introduction to Relational Model-Integrity constants over Relations, Enforcing Integrity constants-Querying Relational Data, Logical Data Base Design- Introduction To views, Destroying and AlteringTables, and Views, Relational Algebra-Selection, Projection, and Set operations- Renaming- Joins-Divisions.RelationalCalculus-TupleRelationalcalculus-DomainRelationalCalculus

……

Relational Model:

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.It is widely used data model for data storage and processing. This model is simple and has a capability to process data with storage efficiency.

Key Terms:

Tables: In this model relations are saved in the form of tables. It has tuples and fields.

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 Ai must have a domain, dom(Ai)

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 923988	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.

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

- tuple has no duplicate value
- Order of tuple can have a different sequence

Importance of Null Value:

- The SQL NULL is the term used to represent a missing value.
- A NULL value in a table is a value in a field that appears to be blank.
- A field with a NULL value is a field with no value.
- It is very important to understand that a NULL value is different than a zero value or a filed that contains space.

Syntax: Create table customers(id int not null,name varchar(20) not null,age int not null,address char(10),salary decimal(18,2));

- Here, NOT NULL signifies that column should always accept an explicit value of the given data type.
- A field with a NULL value is the one that has been left blank during the record creation.

ID	Name	Age	Address	Salary
1	Rajesh	32	Ahmedabad	20000
2	Kalyan	25	Delhi	15000
3	Kaushik	23	Kolkata	20000
4	Kittu	25	Mumbai	35000
5	Samatha	22	Punjab	30000
6	Komali	24	MP	
7	Rani	27	Kerala	

- **Example:**

Select id, name, age, address, salary
from customers
where salary is not null;

Output:

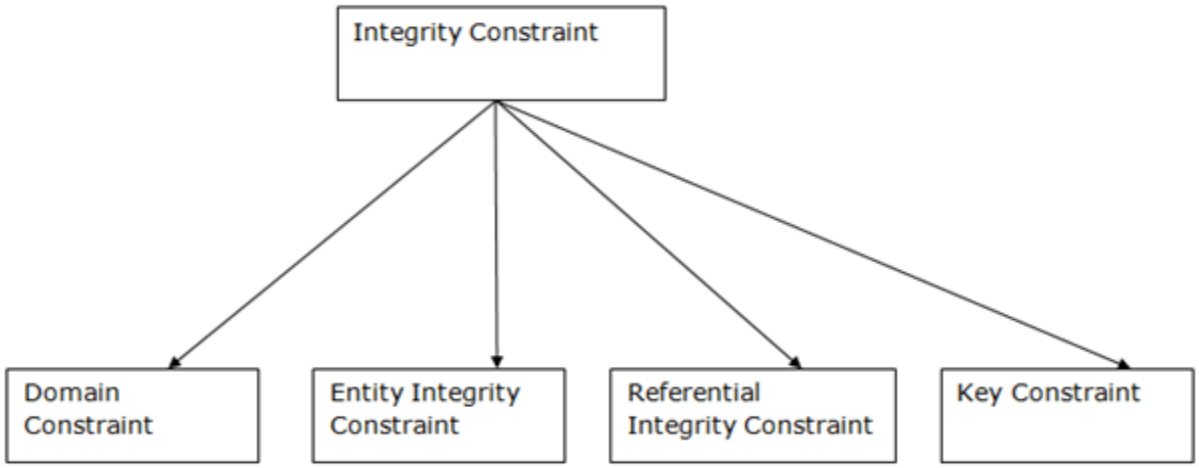
ID	Name	Age	Address	Salary
1	Rajesh	32	Ahmedabad	20000
2	Kalyan	25	Delhi	15000
3	Kaushik	23	Kolkata	20000
4	Kittu	25	Mumbai	35000
5	Samatha	22	Punjab	30000

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

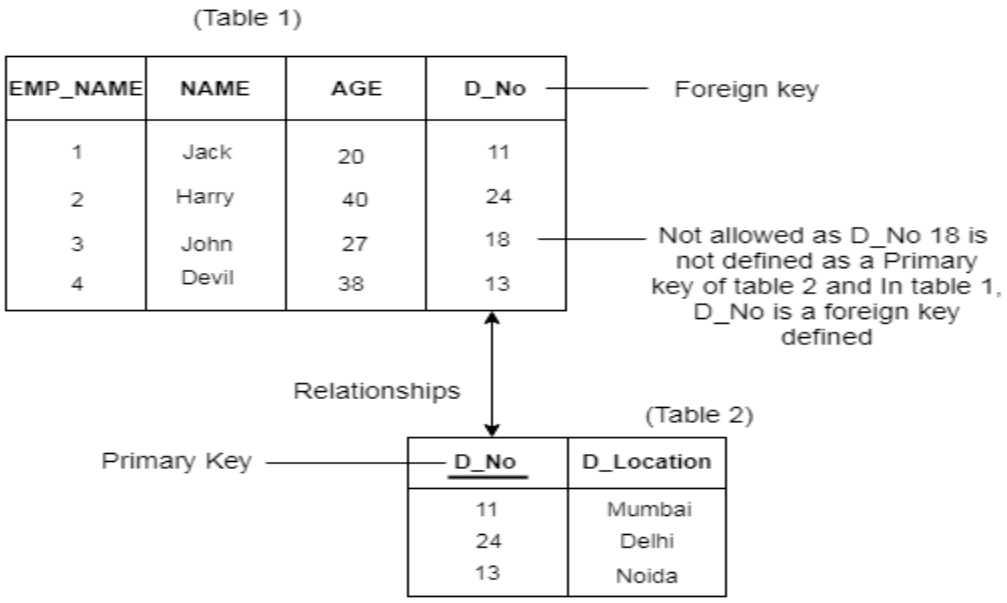
EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

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:



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 value in the relational table.

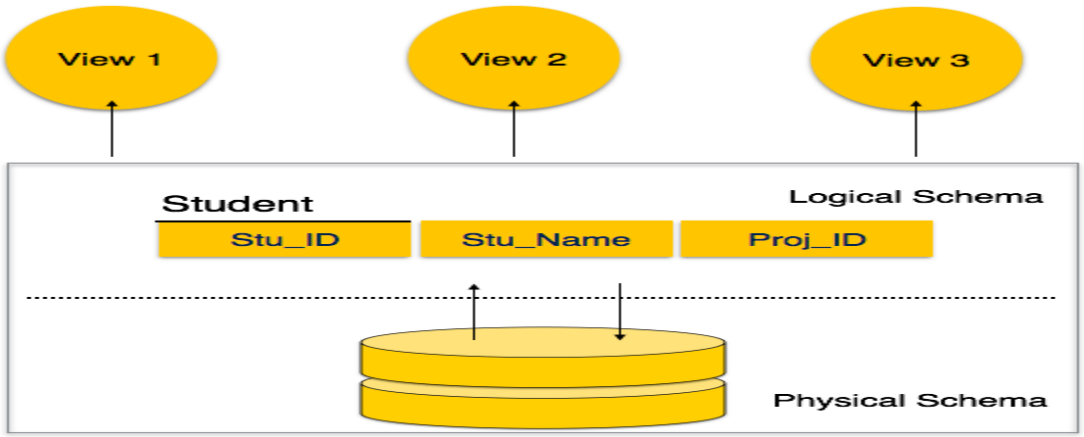
Example:

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

Database Schema (Data Base Design):

- A database schema is the skeleton structure that represents the logical view of the entire database.
- It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.
- A database schema defines its entities and the relationship among them.
- It contains a descriptive detail of the database, which can be depicted by means of schema diagrams.
- It's the database designers who design the schema to help programmers understand the database and make it useful.



A database schema can be divided broadly into two categories.

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

Querying Relational Data:

SQL Data Types:

- Data types are used to represent the nature of the data that can be stored in the database table.

- For example, in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column.

Data types mainly classified into three categories for every database.

1 String Data types
2 Numeric Data types
3 Date and time Data types

1. MySQL String Data Types:

CHAR(Size)	It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1.
VARCHAR(Size)	It is used to specify a variable length string that can contain numbers, letters, and special characters. Its size can be from 0 to 65535 characters.
BINARY(Size)	It is equal to CHAR() but stores binary byte strings. Its size parameter specifies the column length in the bytes. Default is 1.
VARBINARY(Size)	It is equal to VARCHAR() but stores binary byte strings. Its size parameter specifies the maximum column length in bytes.
TEXT(Size)	It holds a string that can contain a maximum length of 255 characters.
TINYTEXT	It holds a string with a maximum length of 255 characters.
MEDIUMTEXT	It holds a string with a maximum length of 16,777,215.
LONGTEXT	It holds a string with a maximum length of 4,294,967,295 characters.
ENUM(val1, val2, val3,...)	It is used when a string object having only one value, chosen from a list of possible values. It contains 65535 values in an ENUM list. If you insert a value that is not in the list, a blank value will be inserted.
SET(val1,val2,val3,....)	It is used to specify a string that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values at one time in a SET list.
BLOB(size)	It is used for BLOBs (Binary Large Objects). It can hold up to 65,535 bytes.

2. MySQL Numeric Data Types:

BIT(Size)	It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1.
INT(size)	It is used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255.
INTEGER(size)	It is equal to INT(size).

FLOAT(size, d)	It is used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by **d** parameter.
FLOAT(p)	It is used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to24, the data type becomes FLOAT (). If p is from 25 to 53, the data type becomes DOUBLE().
DOUBLE(size, d)	It is a normal size floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal is specified by d parameter.
DECIMAL(size, d)	It is used to specify a fixed point number. Its size parameter specifies the total number of digits. The number of digits after the decimal parameter is specified by **d** parameter. The maximum value for the size is 65, and the default value is 10. The maximum value for **d** is 30, and the default value is 0.
DEC(size, d)	It is equal to DECIMAL(size, d).
BOOL	It is used to specify Boolean values true and false. Zero is considered as false, and nonzero values are considered as true.

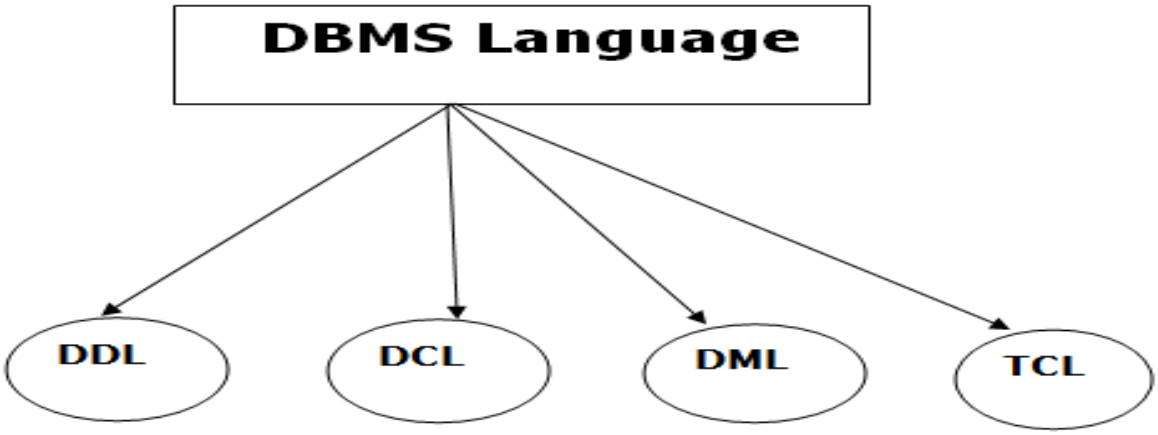
3. MySQL Date and Time Data Types:

DATE	It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'.
TIMESTAMP(fsp)	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
YEAR	It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

SQL Languages:

Database languages can be used to read, store and update the data in the database.

Types of Database Language:



Data Definition Language (DDL):

Here are some tasks that come under DDL:

1. Create: It is used to create objects in the database.
2. Alter: It is used to alter the structure of the database.
3. Drop: It is used to delete objects from the database.
4. Truncate: It is used to remove all records from a table.
5. Rename: It is used to rename an object.

Comment: It is used to comment on the data dictionary.

Data Manipulation Language (DML):

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

Here are some tasks that come under DML:

1. Select: It is used to retrieve data from a database.
2. Insert: It is used to insert data into a table.
3. Update: It is used to update existing data within a table.
4. Delete: It is used to delete all records from a table.

Data Control Language (DCL):

It is used to retrieve the stored or saved data.The DCL execution is transactional. It also has rollback parameters.(But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

1. Grant: It is used to give user access privileges to a database.
2. Revoke: It is used to take back permissions from the user.

Transaction Control Language (TCL):

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

Here are some tasks that come under TCL:

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

SQL Queries:

1. **Create:** It is used to create a new table in a database..

Syntax: create table table_name (column1 datatype,
column2datatype,
column3datatype,....);

Example:CREATE TABLE Persons (PersonIDint,LastNamevarchar(255),FirstNamevarchar(255), Address varchar(255),City varchar(255));

Output:

PersonID	LastName	FirstName	Address	City

2. **Insert:** it is used to insert new records in a table.

Syntax: insert into table_name(col1,clo2,clo3,…colN)
values(value1,value2,value3,…..valueN);

If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table.

Syntax: insert into table_name
values(value1,value2,value3,…..valueN);

Example:INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen21', 'Stavanger', '4006', 'Norway');

Output:

CustomerName	ContactName	Address	City	PostalCode	Country
Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

3. **Drop**: it is used to delete both the structure and records stored in the table.

Syntax: drop table table_name;

Example: drop table student;

4. **Alter:** it is used to add, delete, or modify columns in an existing table and also used to add and drop various constraints on an existing table.
   a) **Syntax for Add Column:** ALTER TABLE table_name
      ADD column_namedatatype;;

Example: alter table student add smarksinteger(20);

   b) **Syntax for Drop Column:** ALTER TABLE table_name
      Drop column column_namedatatype;;

Example: alter table student drop column smarks;

   c) **Syntax for Alter/Modify Column:** ALTER TABLE table_name
      ALTER COLUMN column_namedatatype;

(And)

      ALTER TABLE table_name
      MODIFY COLUMN column_namedatatype;

5. **Rename:** it is used to rename the table.
   Syntax: rename oldtable_name to newtable_name;
   Example: rename student to studcse;

6. **Truncate**: it is used to delete the data inside a table, but not the table itself.
   Syntax: truncate table table_name;
   Example: truncate table student;

7. **Update**: it is used to update modify the existing records in a table.
   Syntax: UPDATE table_name
      SET column1 = value1, column2 = value2, ...
      WHERE condition;

Customers Table:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	AlfredsFutterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico

Example: UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	AlfredsFutterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico

8. **Delete:** it is used to delete existing records in a table.
Syntax: delete from table_name[where condition];
Example: Delete from student where sid=’502’;

9. **Select:** it is used to select select data from a database. The data returned is stored in a result table, called the result-set.
Syntax:SELECT column1, column2, ...
FROM table_name
WHERE condition;
Example: select sid,sname,sdepat
From student
Where sid=’505’;
If you want to select all the fields available in the table
Syntax: Select * from student;

SQL Operators:

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

Generally there are three types of operators in SQL:

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

1. SQL Arithmetic Operators:

Let's assume two variables "a" and "b". Here "a" is valued 50 and "b" valued 100.

Example:

Operators	Descriptions	Examples
+	It is used to add containing values of both operands	a+b will give 150
-	It subtracts left hand operand from Right hand operand	a-b will give -50
*	It multiply both operand's values	a*b will give 5000
/	It divides Right hand operand by left hand operand	b/a will give 2
%	It divides Right hand operand by left hand operand and returns reminder	b%a will give 0

2. SQL Comparison Operators:

Let's take two variables "a" and "b" that are valued 50 and 100.

Operator	Description	Example
=	Examine both operands value that are equal or not,if yes condition become true.	(a=b) is not true
!=	This is used to check the value of both operands equal or not,if not condition become true.	(a!=b) is true
<>	Examines the operand's value equal or not, if values are not equal condition is true	(a<>b) is true
>	Examine the left operand value is greater than right Operand, if yes condition becomes true	(a>b) is not true
<	Examines the left operand value is less than right Operand, if yes condition becomes true	(a<="" td="">
>=	Examines that the value of left operand is greater than or equal to the value of right operand or not,if yes condition become true	(a>=b) is not true
<=	Examines that the value of left operand is less than or equal to the value of right operand or not, if yes condition becomes true	(a<=b) is true
!<	Examines that the left operand value is not less than the right operand value	(a!<="" td="">
!>	Examines that the value of left operand is not greater than the value of right operand	(a!>b) is true

3. SQL Logical Operators:

This is the list of logical operators used in SQL.

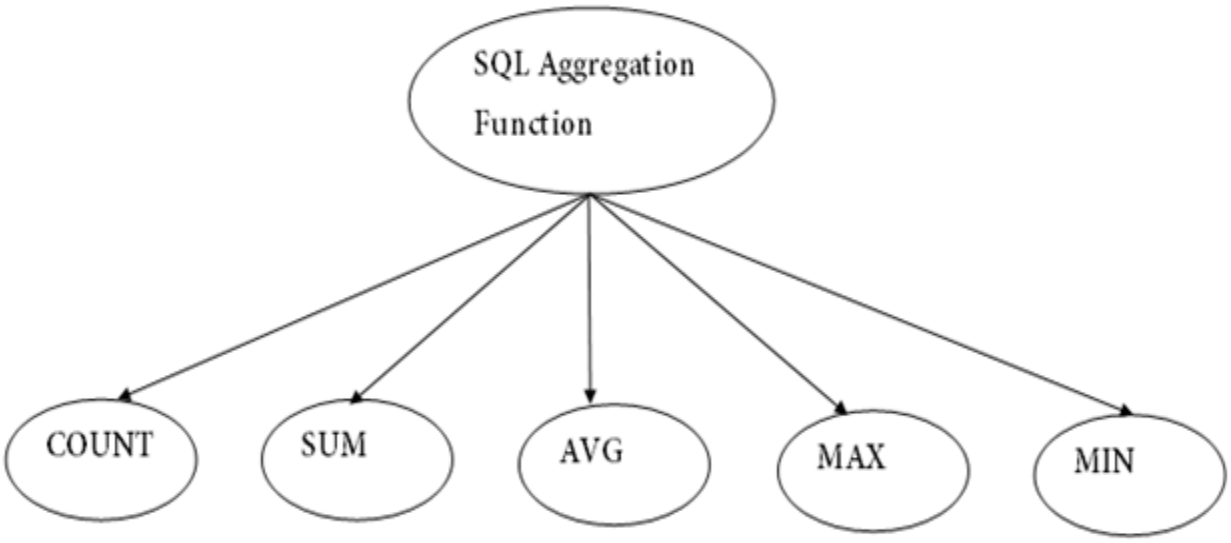
Operator	Description
ALL	to compare a value to all values in another value set.

AND	operator allows the existence of multiple conditions in an SQL statement.
ANY	tocompare a value to any applicable value in the list as per the condition.
BETWEEN	to search for values, that are within a set of values
IN	to compare a value to that specified list value
NOT	reverse the meaning of any logical operator
OR	to combine multiple conditions in SQL statements
EXISTS	to search for the presence of a row in a specified table
LIKE	to compare a value to similar values using wildcard operator

SQL Aggregate Functions:

SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.

Types of SQL Aggregation Function:



1. **COUNT():**

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

Syntax:

COUNT(*)
or
COUNT([ALL|DISTINCT] expression)

Sample table:

PRODUCT_MAST:

PRODUCT	COMPANY	QTY	RATE	COST
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60
Item4	Com3	5	10	50
Item5	Com2	2	20	40
Item6	Com1	3	25	75
Item7	Com1	5	30	150
Item8	Com1	3	10	30
Item9	Com2	2	25	50
Item10	Com3	4	30	120

Example: SELECT COUNT(*)
FROM PRODUCT_MAST;
Output:10
Example: SELECT COUNT(*)
FROM PRODUCT_MAST;
WHERE RATE>=20;
Output:7
Example: SELECT COUNT(DISTINCT COMPANY)
FROM PRODUCT_MAST;
Output:3
Example: SELECT COMPANY, COUNT(*)
FROM PRODUCT_MAST
GROUP BY COMPANY;
Output:
Com1 5
Com2 3
Com3 2
Example: SELECT COMPANY, COUNT(*)
FROM PRODUCT_MAST
GROUP BY COMPANY
HAVING COUNT(*)>2;
Output:
Com1 5
Com2 3

2. SUM():

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

Syntax: SUM()
or
SUM([ALL|DISTINCT] expression)

Example:SELECT SUM(COST)
FROM PRODUCT_MAST;
Output:
670
Example: SELECT SUM(COST)
FROM PRODUCT_MAST
WHERE QTY>3;
Output:320

3. AVG():

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

Syntax:AVG()
or
AVG([ALL|DISTINCT] expression)
Example:SELECT AVG(COST)
FROM PRODUCT_MAST;
Output:67.00

4. MAX():

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

Syntax: MAX()
or
MAX([ALL|DISTINCT] expression)
Example:SELECT MAX(RATE)
FROM PRODUCT_MAST;
Output:30

5. MIN():

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

Syntax: MIN()
or
MIN([ALL|DISTINCT] expression)
Example:SELECT MIN(RATE)
FROM PRODUCT_MAST;
Output:10

Scalar functions:

These functions are based on user input, these too returns single value.

1 UCASE()
2 LCASE()

3 MID()
4 LEN()
5 ROUND()
6 NOW()
7 FORMAT()

Example: Students-Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

1. **UCASE()**: It converts the value of a field to uppercase.

Syntax:

SELECT UCASE(column_name) FROM table_name;

Example:

SELECT UCASE(NAME) FROM Students;

Output:

NAME
HARSH
SURESH
PRATIK
DHANRAJ
RAM

2. **LCASE()**: It converts the value of a field to lowercase.

Syntax:

SELECT LCASE(column_name) FROM table_name;

Example:

SELECT LCASE(NAME) FROM Students;

Output:

NAME
harsh
suresh
pratik
dhanraj
ram

3. **MID():** The MID() function extracts texts from the text field.

Syntax:

SELECT MID(column_name,start,length) AS some_name FROM table_name;

Example:

SELECT MID(NAME,1,4) FROM Students;

Output:

NAME
HARS
SURE
PRAT
DHAN
RAM

4. **LEN():** The LEN() function returns the length of the value in a text field.

Syntax:

SELECT LENGTH(column_name) FROM table_name;

Example:

SELECT LENGTH(NAME) FROM Students;

Output:

NAME
5
6
6
7
3

5. ROUND():

The ROUND() function is used to round a numeric field to the number of decimals specified.

Syntax:

SELECT ROUND(column_name,decimals) FROM table_name;

Example:

SELECT ROUND(MARKS,0) FROM Students;

Output:

MARKS
90
50
80
95
85

6. NOW(): The NOW() function returns the current system date and time.

Syntax:

SELECT NOW() FROM table_name;

Example:

SELECT NAME, NOW() AS DateTime FROM Students;

Output:

NAME	DateTime
HARSH	1/13/2017 1:30:11 PM
SURESH	1/13/2017 1:30:11 PM
PRATIK	1/13/2017 1:30:11 PM
DHANRAJ	1/13/2017 1:30:11 PM
RAM	1/13/2017 1:30:11 PM

7. **FORMAT():** The FORMAT() function is used to format how a field is to be displayed.

Syntax:

SELECT FORMAT(column_name,format) FROM table_name;

Example:

SELECT NAME, FORMAT(Now(),'YYYY-MM-DD') AS Date FROM Students;

Output:

NAME	Date
HARSH	2017-01-13
SURESH	2017-01-13
PRATIK	2017-01-13
DHANRAJ	2017-01-13
RAM	2017-01-13

SQL Date and Time Functions:

1. SYSDATE():

This function returns the current date and time of the system. It is one of the most popular oracle functions. SYSDATE() is popularly used with the function TO_CHAR().

Syntax: select sysdate from dual;

This returns the system date and time in the form of a string. In this case, it will be ' 08-01-2018 12:28:34'.

2. MONTHS_BETWEEN(x,y):

This function takes two values namely x and y which are in the form of months. It returns the number of months between x and y.

Example: SELECT MONTHS_BETWEEN (SYSDATE, EMP_JOIN_DATE)FROM EMP;

Consider the Employee joining date as 1-January-2018 and the system date as 1-August-2018. Therefore the above returns 7.

3. ADD_MONTHS(d,n):

This function gives the same day as d, n number of months away. The value of n can be positive or negative.

Example:SELECT SYSDATE, ADD_MONTHS (SYSDATE,2)FROM DUAL;

This function will return the sysdate and the date 2 months after the sysdate i.e. '1-August-2018' and '1-October-2018'.

4. LAST_DAY(d):

This function returns the last day of the month for the specific month d provided in the function.

Example:SELECT SYSDATE, LAST_DAY (SYSDATE)FROM DUAL;

This returns the system date and the last day of the particular month for the system date i.e. '1-August-2018' and '31-August-2018'.

SQL Numeric Functions:

1. ABS(X):

This function returns the absolute value of X.

Example:SELECT ABS(-10);

This returns 10.

2. MOD(X,Y):

The variable X is divided by Y and their remainder is returned.

Example: SELECT MOD(15,2);

This returns 1.

3. SIGN(X):

This returns 1 if X is positive, -1 if it is negative and 0 if the value of X is 0. For

Example:

SELECT SIGN(-20);

This returns -1.

4. FLOOR(X):

This returns the largest integer value that is either less than X or equal to it.

Example: SELECT FLOOR(8.3);

This returns 8.

5. CEIL(X):

This returns the smallest integer value that is either more than X or equal to it.

Example:SELECT CEIL(8.3);

This returns 9.

6. POWER(X,Y):

This function returns the value of X raised to the power of Y.

Example:SELECT POWER(3,2);

This returns 9.

SQL String Functions:

1. ASCII(str):

This function returns the ASCII or numeric value of the first word in the string str provided. If it is an empty string, it returns 0.

Example: SELECT ASCII('Apple');

This returns the ASCII value of A i.e. 65 as it is the first character in the string.

2. CONCAT(str1,str2…..strn):

This function returns the string that forms by concatenating all the strings in the argument list.These strings may be only two or multiple but they will all be concatenated.

Example:SELECT CONCAT('Sky', 'Is', 'Beautiful');

Three strings 'Sky', 'Is', 'Beautiful' are concatenated into a single string i.e ‘SkyIsBeautiful’

3. LENGTH(str):

This function returns the length of the string str in bytes.

Example:SELECT LENGTH('happy');

The length of the string “happy” is returned in bytes i.e 5.

4. LOWER(str):

All the characters in uppercase are converted to lowercase by this function.

Example:SELECT LOWER('BEAUTY');

All the characters of “BEAUTY” are converted to lowercase i.e “beauty”

5. STRCMP(str1,str2):

This function compares both the strings str1 and str2. It returns 0 if both strings are equal, 1 if str1 is greater than str2 and -1 and if str2 is greater than str1.

Example: SELECT STRCMP('MIKE', 'MIKE');

The function returns 0 as the strings “MIKE” and “MIKE” are identical.

6. UPPER(str):

All the characters in lowercase are converted to uppercase by this function.

Example:SQL> SELECT UPPER('orange');

The string ”orange” is converted to “ORANGE” in uppercase.

SQL String Conversion Functions:

There are 3 types of conversion functions.

1. To_Char
2. To_Number
3. To_Date

1. To-Char():

TO_CHAR function is used to typecast a numeric or date input to character type with a format model (optional).

Syntax: To_Char(n, format) from table_name;

Example: select to_char(sysdate,’dd-mm-yyyy’) from dual;

```
SQL> select to_char(sysdate,'dd-mm-yyyy') from dual;

TO_CHAR(SY
----------
16-04-2021
```

2. To_Number():

The TO_NUMBER function converts a character value to a numeric datatype. If the string being converted contains nonnumeric characters, the function returns an error.

Syntax: To_Number(string, format) from table_name;

Example:SELECT TO_NUMBER('121.23', '9G999D99') FROM DUAL;

3. To_Date():

The function takes character values as input and returns formatted date equivalent of the same.

Syntax: To_Date(characters, format) from table_name;

Example:SELECT TO_DATE('January 15, 1989, 11:00 A.M.', 'Month dd, YYYY, HH:MI A.M.', 'NLS_DATE_LANGUAGE = American')FROM DUAL;

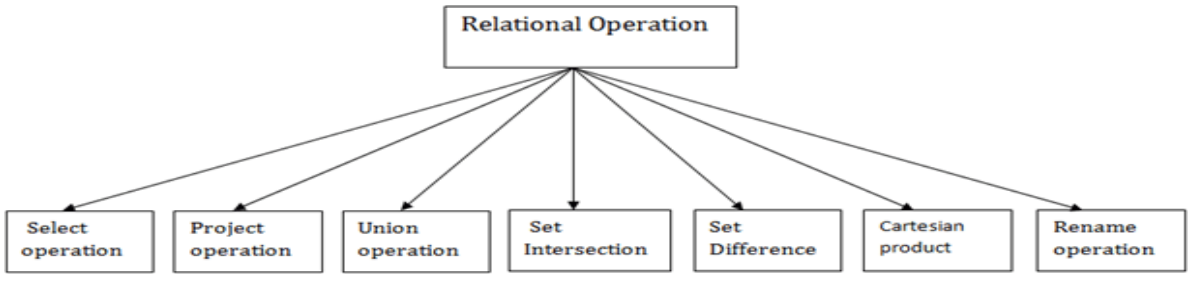
```
SQL> select to_date('17-april-2021','dd-mm-yyyy') from dual;

TO_DATE('
---------
17-APR-21
```

Relational Algebra:

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

Types of Relational operation



Operations:

1. Select Operation:

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

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.

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

Example:

σ BRANCH_NAME="perryride" (LOAN)

Output:

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

2. Project Operation:

a. 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.
b. It is denoted by ∏.

1. Notation:∏A1,A2,A3(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. UnionOperation:
   - 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:

DEPOSITORRELATION

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

BORROWRELATION

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)U∏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 denotedby intersection ∩.

1. Notation: R∩ 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 butnot in S.
- It is denotedby intersection minus(-).

1. Notation:R-S

Example: Using the above DEPOSITOR table and BORROWtable

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 across 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 there name operator to rename STUDENT relation to STUDENT1.

1. ρ(STUDENT1,STUDENT)

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

SALARY

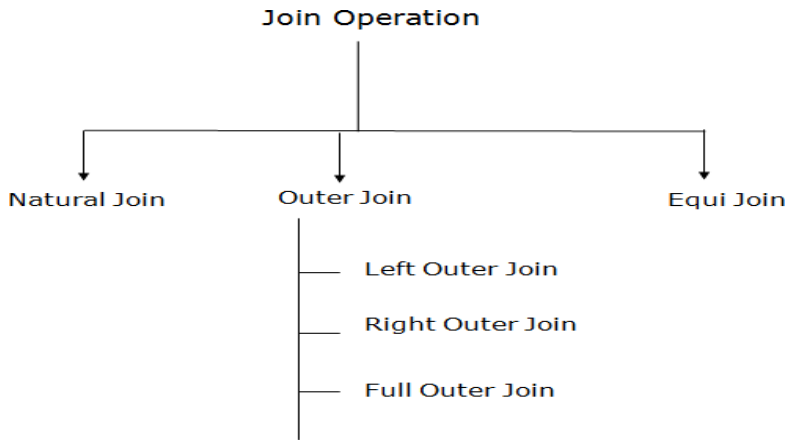
EMP_CODE	SALARY
101	50000
102	30000
103	25000

1. 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 us observe above EMPLOYEE table and SALARY table:

Input:

1. ∏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	Civilline	Mumbai
Shyam	Parkstreet	Kolkata
Ravi	M.G.Street	Delhi
Hari	Nehrunagar	Hyderabad

FACT_WORKERS

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

Input:

1. (EMPLOYEE ⋈ FACT_WORKERS)

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civilline	Mumbai	Infosys	10000
Shyam	Parkstreet	Kolkata	Wipro	20000
Hari	Nehrunagar	Hyderabad	TCS	50000

An outer join is basically of three types:

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

a. Lef touter 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:

1. EMPLOYEE ⟕ FACT_WORKERS

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civilline	Mumbai	Infosys	10000
Shyam	Parkstreet	Kolkata	Wipro	20000
Hari	Nehrustreet	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 table

Input:

1. EMPLOYEE⟖FACT_WORKERS

Output:

EMP_NAME	BRANCH	SALARY	STREET	CITY

Ram	Infosys	10000	Civilline	Mumbai
Shyam	Wipro	20000	Parkstreet	Kolkata
Hari	TCS	50000	Nehrustreet	Hyderabad
Kuber	HCL	30000	NULL	NULL

c. Full outer join:

- Full Join provides result with concatenation of LEFT JOIN and RIGHT JOIN.
- The result will contain all the rows from both Table 1 and Table 2. The rows having no matching in result table will have NULL values.It is denoted by ⟗.

Example: Using the above EMPLOYEE table and FACT_WORKERS table

Input:

1. EMPLOYEE⟗FACT_WORKERS

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civilline	Mumbai	Infosys	10000
Shyam	Parkstreet	Kolkata	Wipro	20000
Hari	Nehrustreet	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:

CUSTOMERRELATION

CLASS_ID	NAME
1	John
2	Harry
3	Jackson

PRODUCT

PRODUCT_ID	CITY
1	Delhi
2	Mumbai
3	Noida

Input:

1. CUSTOMER⋈PRODUCT

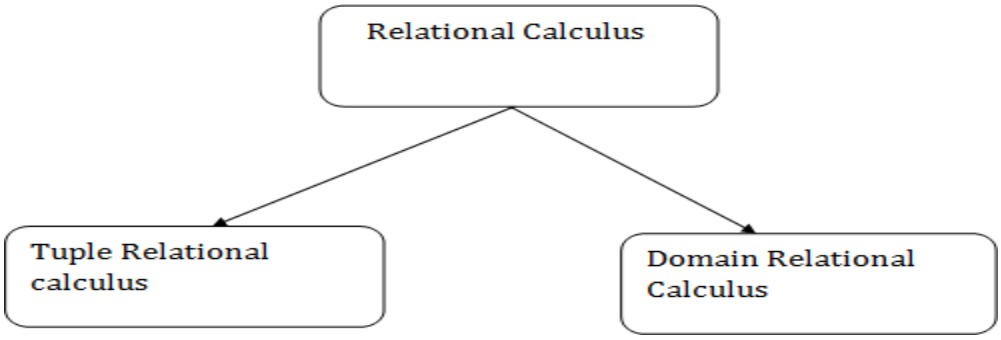
Output:

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

RelationalCalculus

- 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)ANDT.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'.

TRC (tuple relation calculus) can be quantified. In TRC, we can use Existential (∃) and UniversalQuantifiers(∀).

Forexample:

1.{R|∃T∈Authors(T.article='database'ANDR.name=T.name)}

Output: This query will yield the same result as the previous one.

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 >|∈javatpoint ∧ subject ='database'}

Output: This query will yield the article,pageand subject from the relational javatpoint,where the subject is a database.