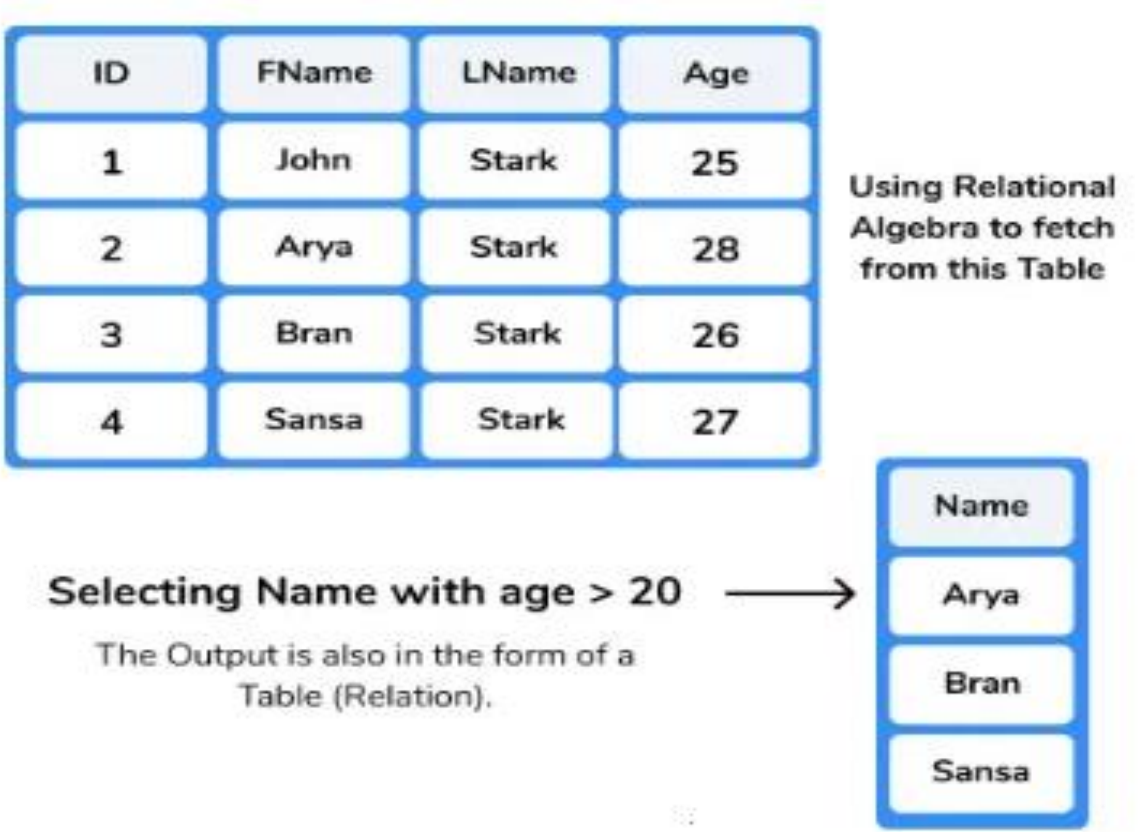
Unit – 3

The Relational Algebra and Relational Calculus

Relational Algebra

- Every database management system must define a query language to allow user to access the data stored in the database.
- Relational Algebra is a procedural query language used to query the database tables to access data in different ways.
- In relational algebra, input is a relation (table from which data has to be accessed) and output is also a relation (a temporary table holding the data asked for the by the user).
- The primary operations that we can perform using relational algebra are:
  - Select, Project, Union, Set Difference, Cartesian Product and Join.

Relational Algebra



ID	FName	LName	Age
1	John	Stark	25
2	Arya	Stark	28
3	Bran	Stark	26
4	Sansa	Stark	27

Name
Arya
Bran
Sansa

Relational Algebra

We can use Relational Algebra to fetch data from this Table(relation)

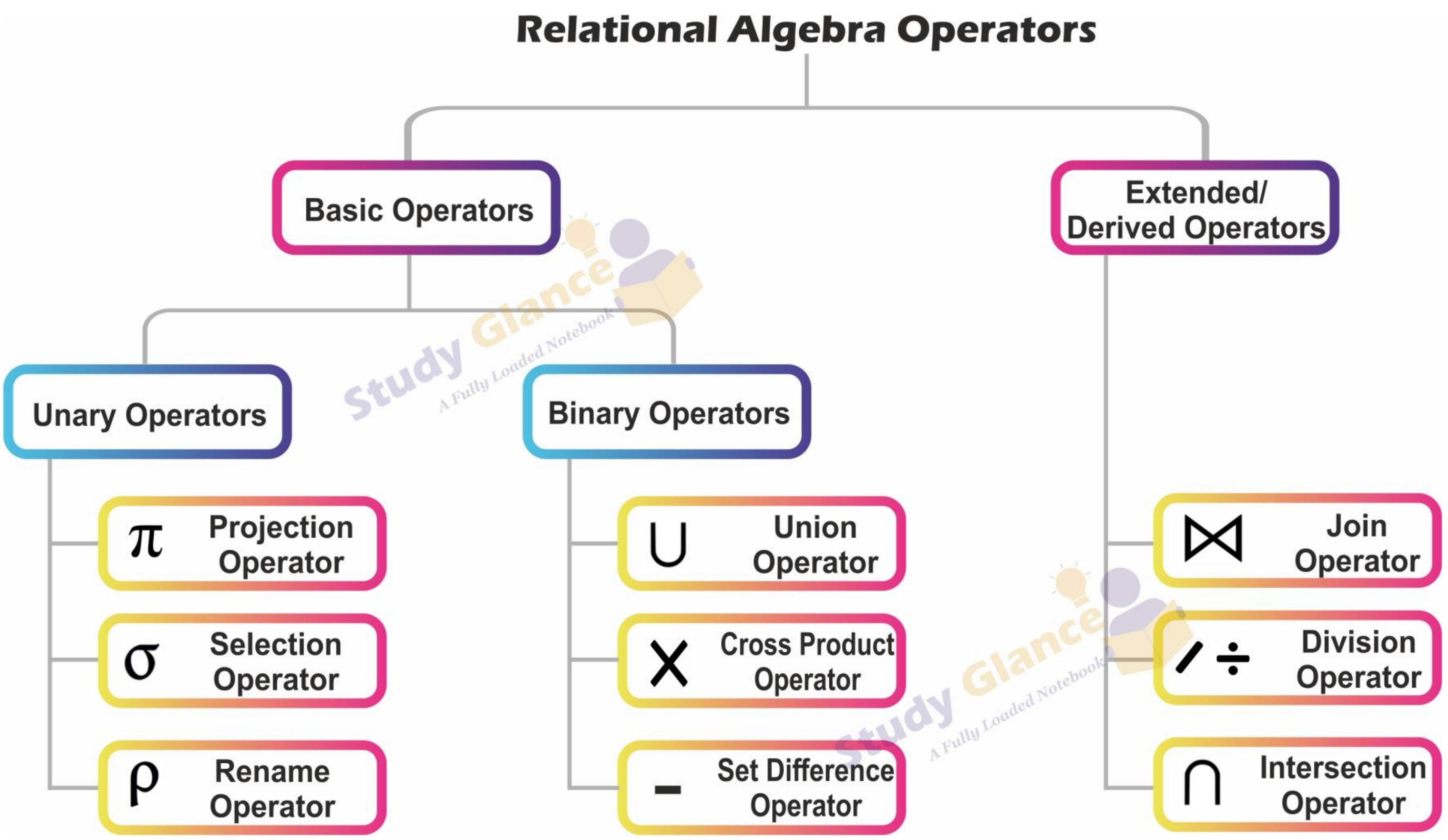
ID	Name	Age
1	Akon	17
2	Bkon	19
3	Ckon	15
4	Dkon	13

Select Name students with age less than 17

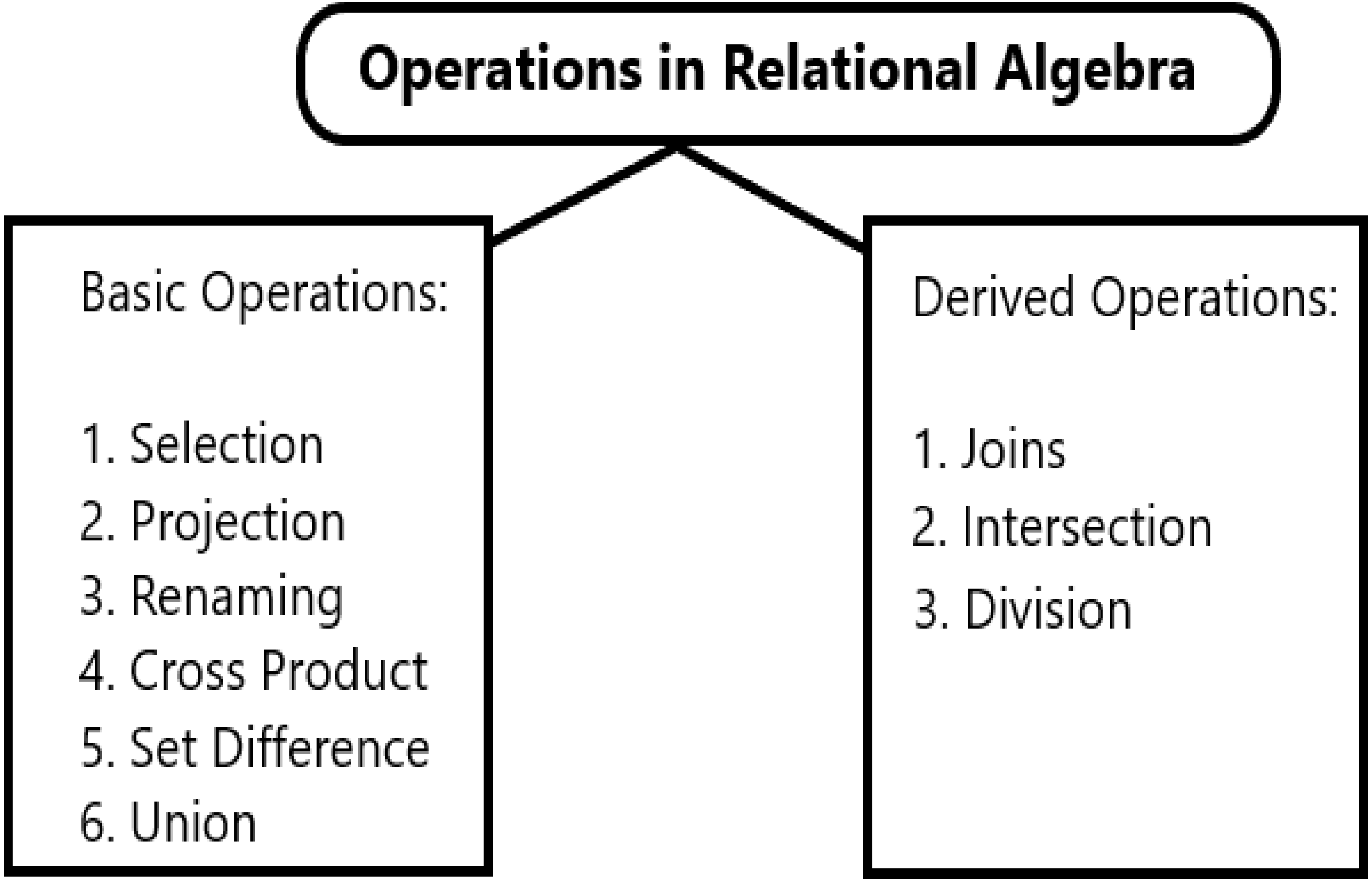
Output

Name
Ckon
Dkon

The output for query is also in form of a table(relation), with results in different columns



Relational Algebra Operators
Basic Operators
Extended/ Derived Operators
Unary Operators
Binary Operators
π Projection Operator
σ Selection Operator
ρ Rename Operator
∪ Union Operator
X Cross Product Operator
− Set Difference Operator
⋈ Join Operator
/ ÷ Division Operator
∩ Intersection Operator


Operations in Relational Algebra
Basic Operations:
1. Selection
2. Projection
3. Renaming
4. Cross Product
5. Set Difference
6. Union
Derived Operations:
1. Joins
2. Intersection
3. Division

Unary Relational Operators

1. The Select Operation : This operation is used to fetch rows from given table or relation on the basis of given conditions, it is denoted by “Sigma(σ)”.

Syntax : σ $_{\text{<Condition>}}$ (Relation Name)

Here, “σ” is the select operation symbol. R is the relation from which the data needs to be fetched on the basis of conditions. Also, relational operators such as =, <, > etc. can also be used along.

Select Operation Examples:

- **Select tuples from a relation "Books" where subject is "database"**

Solution:

$$\sigma_{subject\,=\,\text{“database”}}(Books)$$

- **Select tuples from a relation "Books" where subject is "database" and price is "450"**

Solution:

$$\sigma_{subject\,=\,\text{“database”}\,\wedge\,price\,=\,\text{“450”}}(Books)$$

- **Select tuples from a relation "Books" where subject is "database" and price is "450" or have a publication year after 2010**

Solution:

$$\sigma_{subject\,=\,\text{“database”}\,\wedge\,price\,=\,\text{“450”}\,\vee\,year\,>\,\text{“ 2010”}}(Books)$$

2. The Project Operation

- This operation is also used to fetch all the rows/tuples/data according to the requested attribute.
- Duplicate tuples are not allowed in the result of a projection operation.
- It means, using project operation one can simply fetch all the tuples corresponding to a single attribute or multiple attributes.
- It does not support any conditions as select operation and is denoted using “Pie(π)”.

Syntax : $\pi_{<attribute>}$**(Relation Name)**

For example : Consider the table of relation R(Roll No, Name, Age, Marks). If we want to project the marks column, then it can be done by :

Query Used : π_{Marks}**(Student_Details)**

Student_Details

Roll No	Name	Age	Marks
1	Anoop	22	30
2	Anurag	23	32
3	Ganesh	21	31

Query Output

Marks
30
32
31

Relational Algebra : Project Operation

Note:
Projection means choosing which columns (or expressions) the query shall return. (Vertical Partitioning)
Selection means which rows are to be returned. (Horizontal Partitioning)

Write relational algebra query for the following questions, consider table student(id, name,address, marks)

Select the name of the student whose id is 3.

Solution:

$$\pi_{name}(\sigma_{id=3}(student))$$

Select id, name and marks of all the students whose id is less than 10 and address is ghorahi.

Solution:

$$\pi_{id,name,marks}(\sigma_{id<10 \land address='ghorahi'}(student))$$

3. The Rename Operation:

- When operations like project and select are performed to fetch new results, these results requires renaming.
- They can be renamed using the rename operation which is denoted using Greek letter “Rho(ρ)”.

Syntax : $\rho_{<New\ Name>}(Relation)$

- **Rename a relation**
  - ✓ Suppose we have a relation named Students and we want to change it to StudentList, the rename operation works as follows:

$$\rho_{\text{StudentList}}(\text{Students})$$

- **Rename an attribute**
  - ✓ Suppose we have a relation named Students and we want to change its attributes StudentID, StudentName to SID and SName, the rename operation works as follows:

$$\rho_{(\text{SID},\text{SName})}(\text{Students})$$

- **Rename both**
  - ✓ Next, we'll change both the relation name and attributes of the Students class:

$$\rho_{\text{StudentList}(\text{SID},\text{SName})}(\text{Students})$$

Sequence of operations

- Selection (σ): Filter rows based on a condition.
- Projection (π): Select specific columns.
- Union (∪): Combine rows from two relations.
- Difference (−): Find rows in one relation but not in another.
- Intersection (∩): Find common rows in two relations.
- Cartesian Product (×): Combine all rows from two relations.
- Join (⋈): Combine rows from two relations based on a common attribute.
- Division (÷): Find rows in one relation that are associated with all rows in another.

✓ When constructing a query, operations are often combined in sequences to achieve the desired result.

✓ Example, filtering rows first, then selecting columns, or joining tables before performing set operations like union or difference.

Relational Algebra Operations from Set Theory

- The Set Theory operations are the standard mathematical operations on set.
- These operations are Binary operations that are, operated on 2 relations unlike PROJECT, SELECT and RENAME operations.
- These operations are used to merge 2 sets in various ways.
- The set operation is mainly categorized into the following:
  - ✓ Union operation
  - ✓ Intersection operation
  - ✓ Set difference or Minus operation
  - ✓ Cartesian Product

Condition For Using Set Theory Operators

To use set theory operators on two relations,

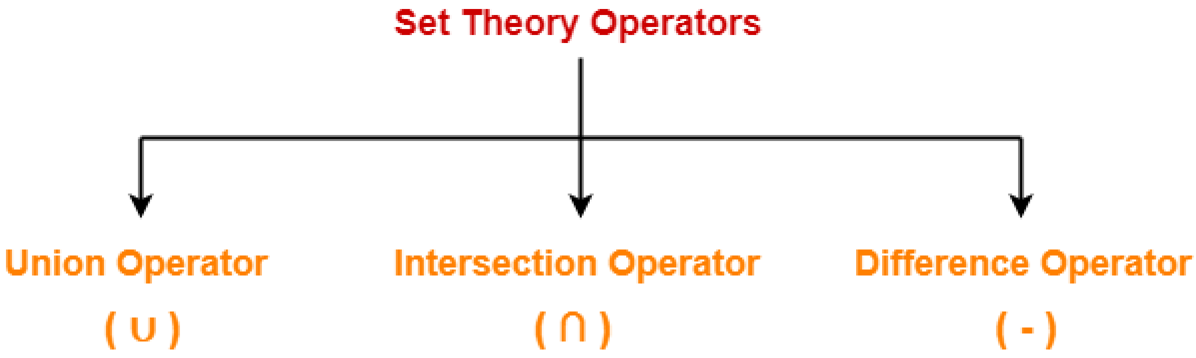
The two relations must be union compatible.

Note:

Union compatible property means:

- Both the relations must have same number of attributes.
- The attribute domains (types of values accepted by attributes) of both the relations must be compatible.

Following operators are called as set theory operators:



1. The Union Operation:

- In order to fetch data from two relations to generate new relation with combined capabilities, union operations can be used.
- The union operation fetches the data from both tables and projects it accordingly. It is denoted through “Union Symbol(U)”.
- Also, two things need to keep in mind while applying union operation are :
  - **Both the relations compulsory to have same number of attributes.**
  - **Both the relations compulsory to have same domain for attributes.**
- Additionally, Duplicate tuples should be automatically removed.

Syntax : X_1 U X_2 , where X_1 & X_2 are two different relations satisfying the above two conditions.

For example : Consider the two tables with relations X_1(Name, Age) and X_2(Name, Age). If we wish to apply the union operation, then it can be done by :

X1(Name, Age)

Name	Age
Anoop	**22**
Anurag	**23**

$\cup$

X2(Name, Age)

Name	Age
Ganesh	21
Saurav	22
Rakesh	20

→

Query Output(X1 U X2)

Name	Age
Anoop	22
Anurag	23
Ganesh	21
Saurav	22
Rakesh	20

Relational Algebra : Union Operation

2. The Intersection Operation

- The intersection operator gives the common data values between the two data sets that are intersected.
- The two data sets that are intersected should be similar for the intersection operator to work.
- Intersection also removes all duplicates before displaying the result.
- It is denoted by $\cap$.

Syntax : $X_1 \cap X_2$, where X_1 & X_2 are two different relations satisfying the above two conditions.

For example : **Consider the two tables with two relations Course_1 and Course_2. If we wish to apply the intersection operation, then it can be done as**

Course_1

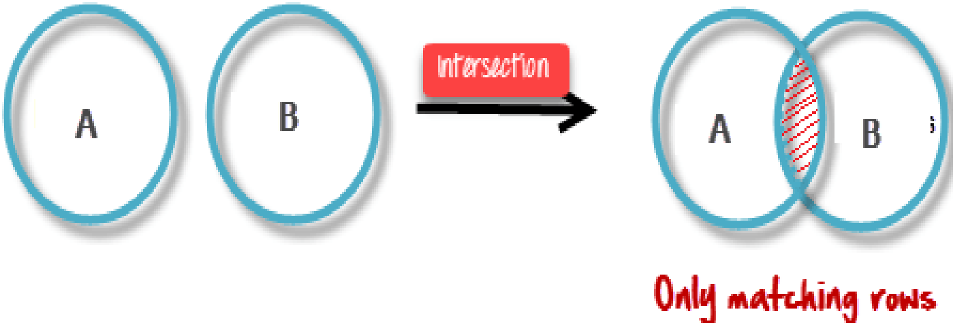
C_id	C_name
11	Foundation C
21	C++
31	JAVA

Course_2

C_id	C_name
12	Python
21	C++

Course_1 ∩ Course_2

C_id	C_name
21	C++



3. The Set Difference Operation:

- In order to fetch the data which is not present in any one of the relation, set difference operation is used.
- The set difference operation is denoted by “Minus(-)”.

Syntax : $X_1 - X_2$ or $X_2 - X_1$, where X_1 & X_2 are two different relations having some attributes

For example : Consider the two tables with relations X_1(Name, Age) and X_2(Name, Age). If we wish to apply the set difference operation, then it can be done by :

X1(Name, Age)

Name	Age
Anoop	22
Saurav	22
Rakesh	20
Pritesh	19

—

X2(Name, Age)

Name	Age
Anoop	22
Anurag	23
Ganesh	21
Saurav	22
Rakesh	20

Query Output(X1 - X2)

Name	Age
Pritesh	19

Query Output(X2 - X1)

Name	Age
Anurag	23
Ganesh	21

Relational Algebra : Set Difference Operation

4. The Cartesian Operation:

- The Cartesian product operation will generate the possible combinations among the tuples from the relations resulting in table containing all the data.
- It combines the information of two or more relations in one single relation. Cartesian product is different from union operation and is denoted by “Cross(X)”.

Syntax : A_1 x A_2 , where A_1 & A_2 are two different relations having some attributes.

For example :

A:

a1	a2
1	2
3	4
5	6

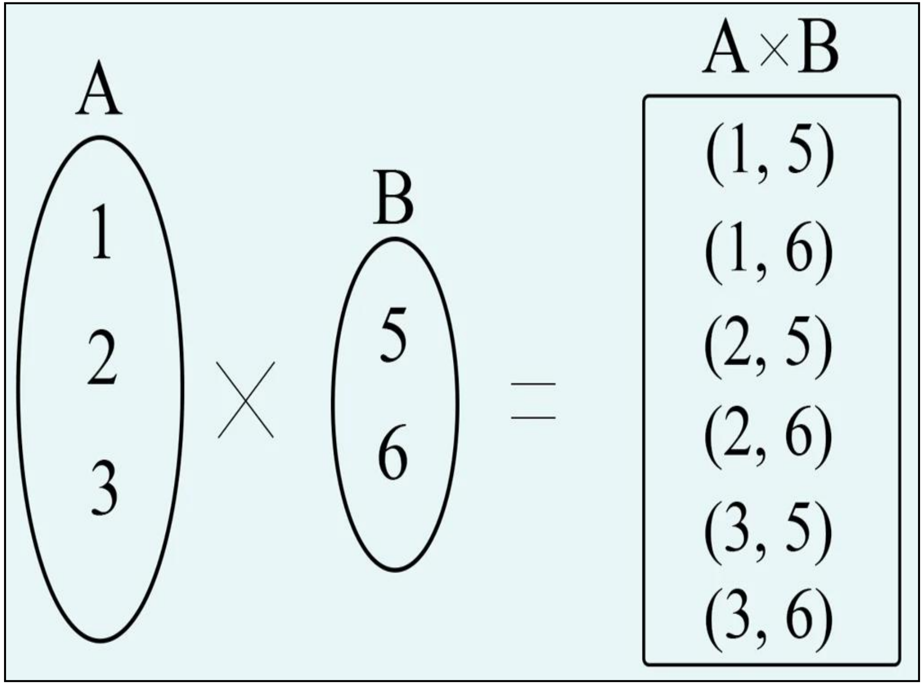
B:

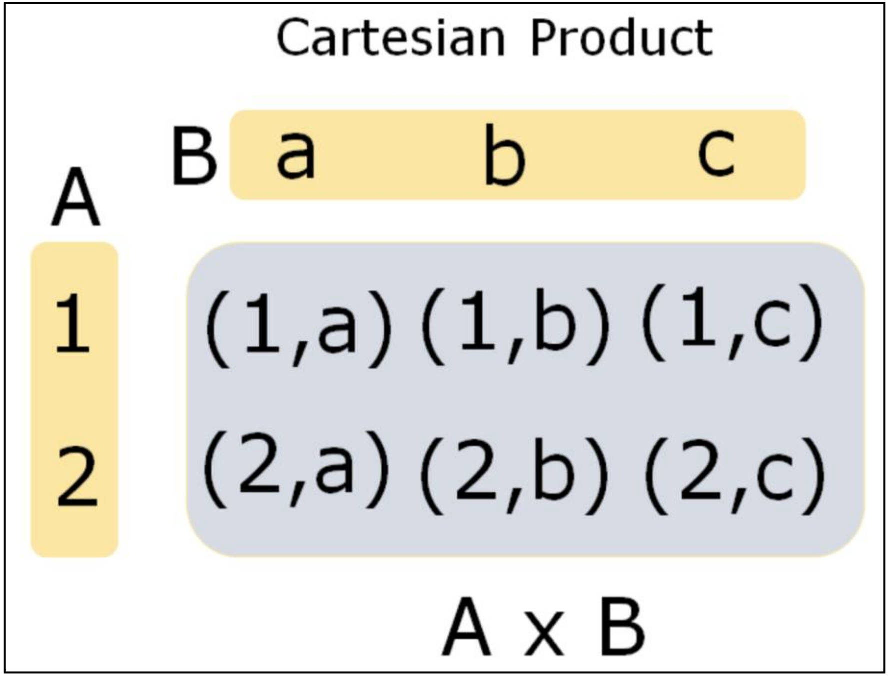
b1
7
8

A × B ⇨

a1	a2	b1
1	2	7
1	2	8
3	4	7
3	4	8
5	6	7
5	6	8

More examples of how cartesian product works





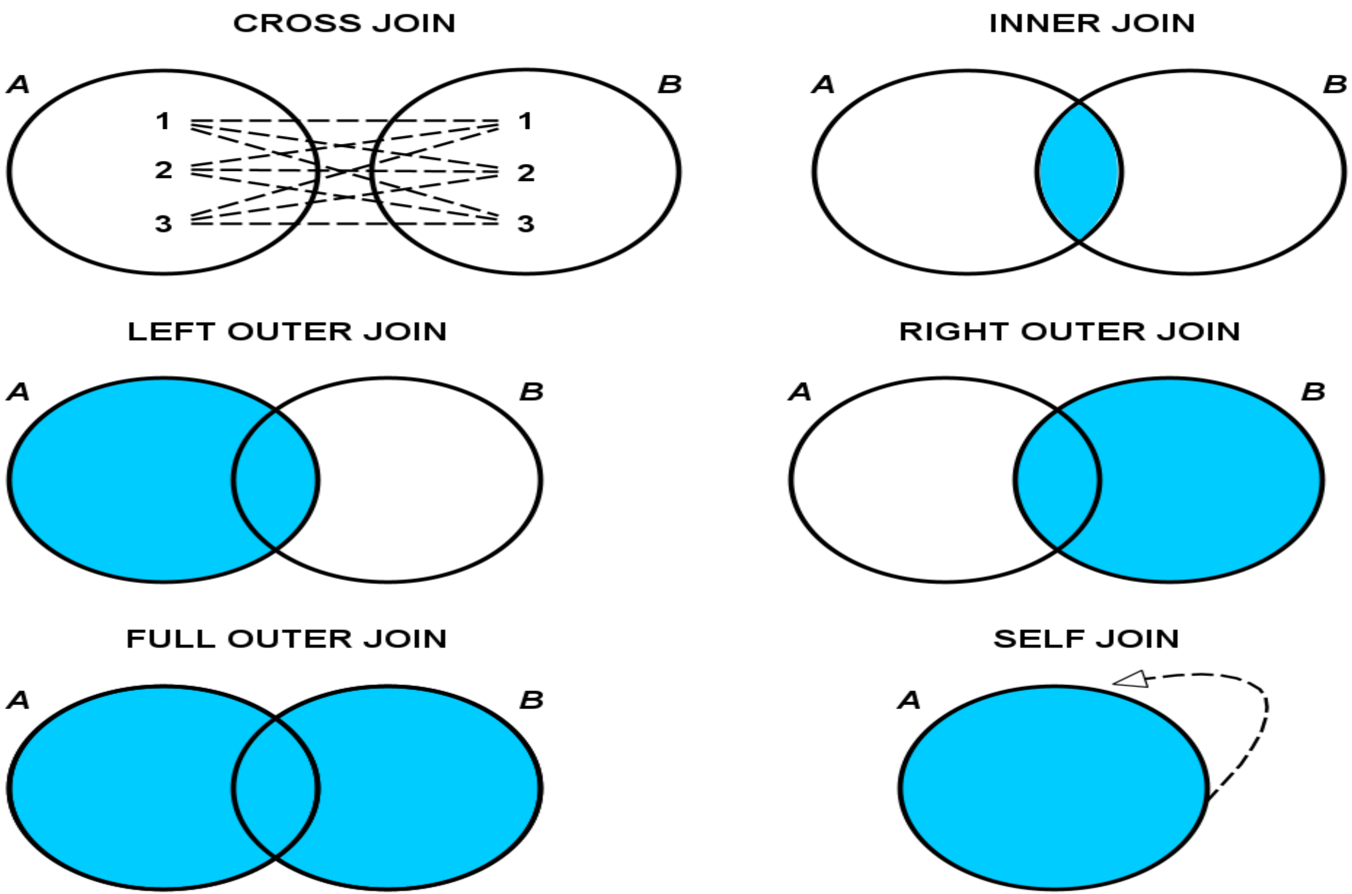
Binary Relational Operations

- In database management systems, the ability to connect and retrieve data from multiple tables is crucial for effective data organization and manipulation.
- The JOIN and DIVISION operations are two binary relational operations that allow users to combine or divide data from multiple tables based on specified conditions.

Join Operation

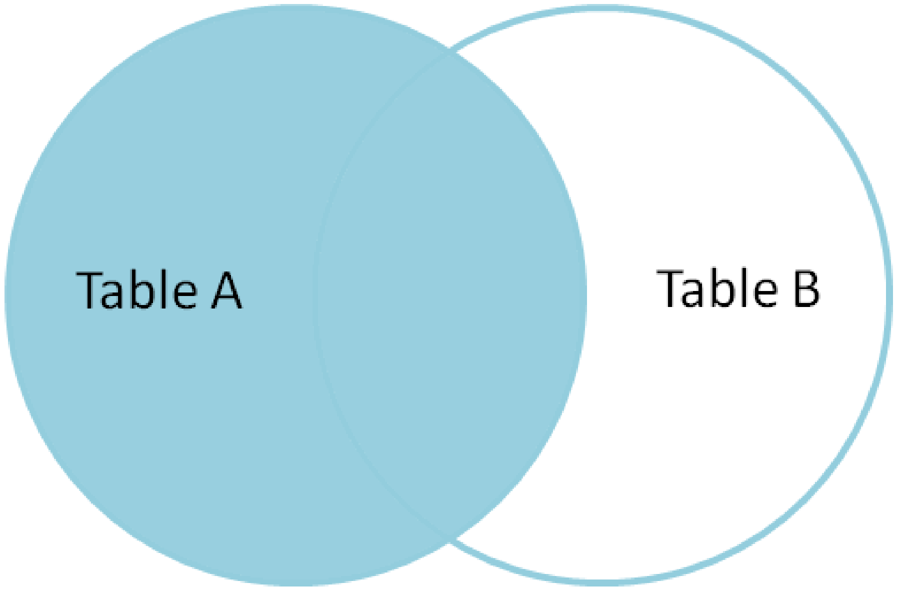
- The goal of creating a join condition is that it helps to combine the data from two or more DBMS tables.
- The tables in DBMS are associated using the primary key and foreign keys.
- Join operation combines the relation R1 and R2 with respect to a condition. It is denoted by $\bowtie$.

Types of Joins



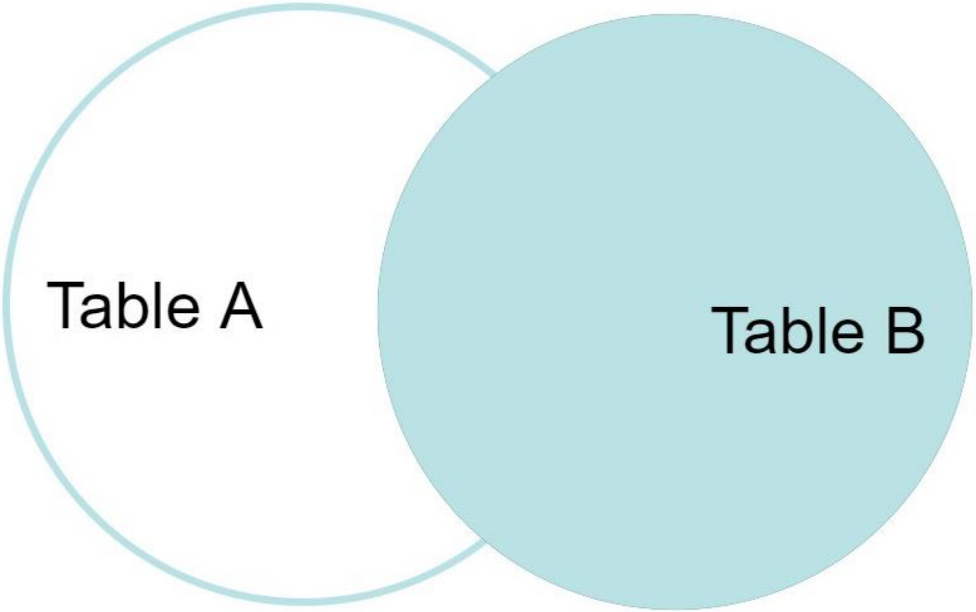
Left Outer Join

- SQL left outer join is also known as SQL left join.
- Suppose, we want to join two tables: A and B. SQL left outer join returns all rows in the left table (A) and all the matching rows found in the right table (B).
- It means the result of the SQL left join always contains the rows in the left table.



Right Outer Join

- SQL right outer join returns all rows in the right table and all the matching rows found in the left table.

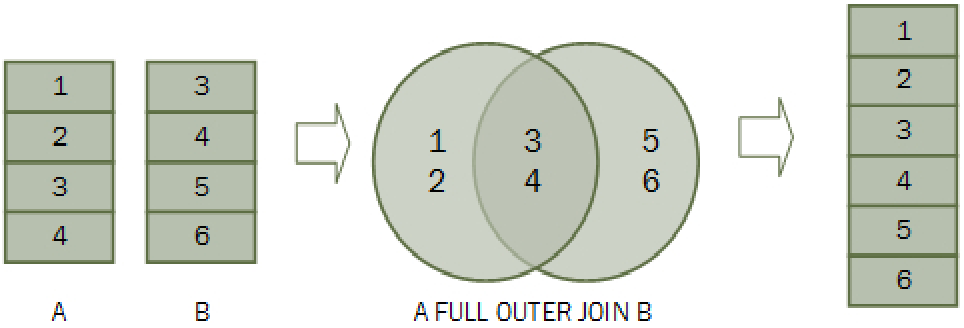


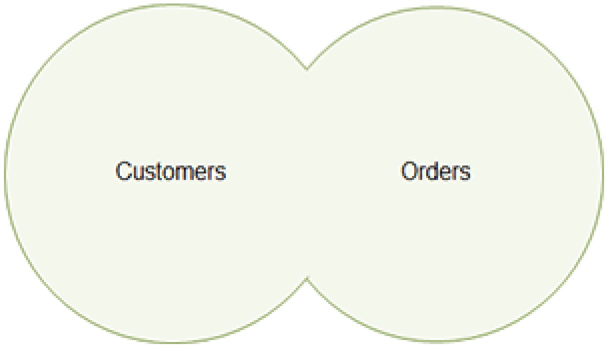
Full Outer Join

SQL full outer join returns:

- all rows in the left table table_A.
- all rows in the right table table_B.
- and all matching rows in both tables.

Some database management systems do not support SQL full outer join syntax e.g., MySQL. Because SQL full outer join returns a result set that is a combined result of both SQL left join and SQL right join.





Natural Join

- The join operation which is used to merge two tables depending on their same column name and data types is known as natural join.
- Natural join can only be performed if there is a common attribute between the relations. The name and type of the attribute must be same.

ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18
7	Salt n Shake	Pcs	-

COMPANY_ID	COMPANY_NAME	COMPANY_CITY
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

** Same column came once

COMPANY_ID	ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_NAME	COMPANY_CITY
16	1	Chex Mix	Pcs	Akas Foods	Delhi
15	6	Cheez-It	Pcs	Jack Hill Ltd	London
15	2	BN Biscuit	Pcs	Jack Hill Ltd	London
17	3	Mighty Munch	Pcs	Foodies.	London
15	4	Pot Rice	Pcs	Jack Hill Ltd	London
18	5	Jaffa Cakes	Pcs	Order All	Boston

Natural Join Example

Division

The division operator is used for queries which involve the 'all'.
R1 ÷ R2 = tuples of R1 associated with all tuples of R2.
Example
Retrieve the name of the subject that is taught in all courses.

Name	Course
System	Btech
Database	Mtech
Database	Btech
Algebra	Btech

÷

Course
Btech
Mtech

=

Name
Database

The resulting operation must have all combinations of tuples of relation S that are present in the first relation or R.

Concept of generalized projection

- It extends the projection operation by allowing arithmetic functions to be used in the projection list.
- It is an enhanced version of project operation which allows to write operations containing attribute names and constants in projection list.
- Example:

$$\Pi_{marks=marks-10}(student)$$

$$\Pi_{average=avg(marks)}(student)$$

Generalized Operation Examples:

- **Find the name and salary of the all employees by increasing their salary by 15%.**

Solution:

$$\Pi_{\text{name, salary=salary*0.15}} \text{(Employee)}$$

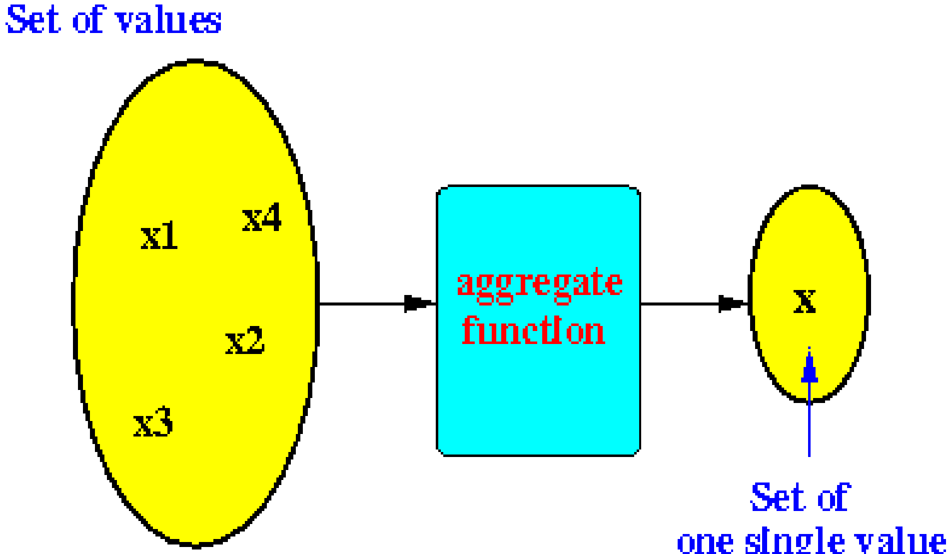
- **Increase the salary of all employees whose age is greater than 20 by 5%.**

Solution:

$$\Pi_{\text{eid, name, age, salary=salary*0.05}(\sigma_{\text{age>20}})} \text{(Employee)}$$

Aggregate Functions

- ✓ In relational algebra, aggregate functions are operations that perform calculations on sets of values and return a single value summarizing those sets.
- ✓ It is denoted by calligraphic G.
- ✓ The most used aggregate functions are:
    - Sum
    - Avg
    - Max
    - Min
    - Count



Examples of Aggregate Functions in Relational Operations

"Find the total amount owed to the credit company."

$$\mathcal{G}_{\textbf{sum}(balance)}(credit\_acct)$$

4275

cred_id	limit	balance
C-273	2500	150
C-291	750	600
C-304	15000	3500
C-313	300	25

credit_acct

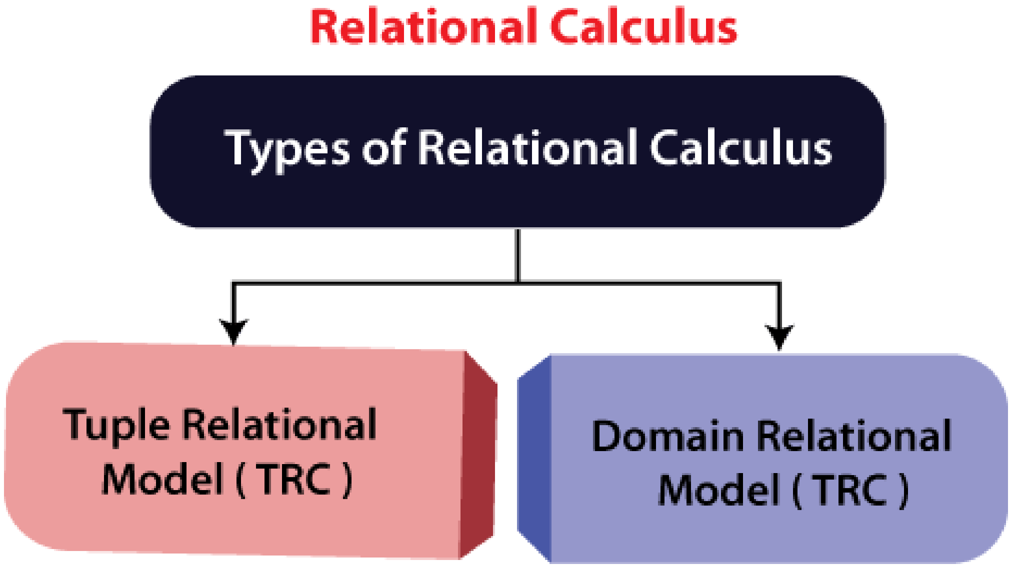
"Find the maximum available credit of any account."

$$\mathcal{G}_{\textbf{max}(available\_credit)}(\Pi_{(limit - balance)\ \textbf{as}\ available\_credit}(credit\_acct))$$

11500

Relational Calculus

- Relational calculus is a formal method used in relational database theory to define queries for retrieving data from a relational database.
- It provides a theoretical foundation for querying databases based on predicate logic and set theory.
- There are basically two types of relational calculus:



Tuple Relational Calculus

- Tuple Relational Calculus (TRC) is a formal language used in relational database theory for expressing queries.
- TRC is a declarative language, meaning that it specifies what data is required from the database, rather than how to retrieve it.
- TRC operates on sets of tuples, which are rows in a relational database table.
- It is primarily used in theoretical contexts to provides valuable insights into database query.
- **Syntax:** The basic syntax of TRC is as follows:

  { t | P(t) } where t is a tuple variable and P(t) is a logical formula that describes the conditions that the tuples in the result must satisfy. The curly braces {} are used to indicate that the expression is a set of tuples

Example 1: let’s say we have a table called “Employees” with the following attributes:

Employee ID
Name
Salary
Department ID

To retrieve the names of all employees who earn more than $50,000 per year, we can use the following TRC query:

$$\{ t \mid \text{Employees}(t) \land t.\text{Salary} > 50000 \}$$

Example 2:

$$\{ T.\text{name} \mid \text{Author}(T) \text{ AND } T.\text{article} = \text{'database'} \}$$

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

Domain Relational Calculus

- It defines queries by specifying a range variable (representing a tuple) and a predicate involving attributes of the tuple.
- 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 $\land$ (and), $\lor$ (or) and $\neg$ (not).
- Notation:

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

  - Where, a1, a2 are attributes

    P stands for formula built by inner attributes
- **Example:**

  {< article, page, subject > | $\in$ javatpoint $\land$ subject = 'database'}

  - This query will yield the article, page, and subject from the relational javatpoint, where the subject is a database.

End!