Relational Algebra

Ram Datta Bhatta

Relational Algebra

- Relational Algebra is a procedural query language, which takes relation as input and generate relation as output. Relational Algebra mainly provides theoretical foundation for relational databases and SQL.
- Relational Algebra is a language for expressing relational database queries.
- It uses operators to perform queries. And operator can be either unary or binary.

Types of operator in Relational Algebra

- **Basic/ fundamental operators**
- **Additional / Derived operators**

Basic/ Fundamental Operators

Selection (σ) :

- selection operator is a unary operator in relational algebra that performs a selection operation .
- It selects tuples (or rows) that satisfy the given condition from a relation.
- It is denoted by (**σ)**

Notation:

$\sigma_{condition}$ (Relation Name)

Example: select tuple from student table whose age is greater than 17

$\sigma_{age>17}$ (Student)

Basic/ Fundamental Operators

Table: student

Roll_no	Name	Age	Address
1	A	20	Bhopal
2	B	17	Mumbai
3	C	16	Mumbai
4	D	19	Delhi
5	E	18	Delhi

Example: select tuple from student table whose age is greater than 17

$\sigma_{age>17}$ (Student)

Basic/ Fundamental Operators

Example: select tuple from student table whose age is greater than 17

$\sigma_{Age > 17}(Student)$

output

Roll_no	Name	Age	Address
1	A	20	Bhopal
4	D	19	Delhi
5	E	18	Delhi

Basic/ Fundamental Operators

Example: select student whose Roll_no is 2.

$\sigma_{Roll\_no = 2}$ (Student)

Roll_no	Name	Age	Address
2	B	17	Mumbai

Basic/ Fundamental Operators

Example: select student Name whose name is D.

σ Name = “D” (Student)

Roll_no	Name	Age	Address
4	D	19	Delhi

Basic/ Fundamental Operators

Example: select tuple from student table whose age is greater than 17 and who lives in Delhi.

σ Age > 17 ∧ Address = “Delhi (Student)

Roll_no	Name	Age	Address
4	D	19	Delhi
5	E	18	Delhi

Basic/ Fundamental Operators

Projection (Π) :

- Projection operator(**Π**) is a unary operator in relational algebra that performs a projection operation .
- It projects (or displays) the particular columns (or attributes) from a relation.
- It delete the column(s) that are not in the projection list.
- It is denoted by (**Π**)

Notation:

Π $_{\text{attribute\_list}}$ (Relation Name)

Duplicate rows are automatically eliminated from result.

Basic/ Fundamental Operators

Example: display the columns roll_no and name from the relation student.

$\Pi_{\text{roll-no, name}}$(student)

Table: student

Roll_no	Name	Age
1	A	20
2	B	17
3	C	18
4	D	19
5	E	18
6	F	18

Output

Roll_no	Name
1	A
2	B
3	C
4	D
5	E
6	F

Basic/ Fundamental Operators

Example: display (Project) the name of students from the relation student.

Π_{name}(student)

Table: student

Roll_no	Name	Age
1	A	20
2	B	17
3	C	18
4	D	19
5	E	18
6	F	18

Output

Name
A
B
C
D
E
F

Basic/ Fundamental Operators

Cartesian Product/Cross Product :

- Cartesian Product/Cross Product is fundamental Operator in relational algebra.
- Cartesian Product combines information of two different relations into one.
- **Notations : R1 X R2**

Basic/ Fundamental Operators

Example:

R1

A	B
1	X
2	Y

R2

C	D
2	P
3	Q

A	B	C	D
1	X	2	P
1	X	3	Q
2	Y	2	P
2	Y	3	Q

Basic/ Fundamental Operators

Example:

R1 X R2

A	B	C	D
1	X	2	P
1	X	3	Q
2	Y	2	P
2	Y	3	Q

$\sigma_{A=C}$ (R1 X R2)

Output

A	B	C	D
2	Y	2	P

Basic/ Fundamental Operators

Example:

R1 X R2

A	B	C	D
1	X	2	P
1	X	3	Q
2	Y	2	P
2	Y	3	Q

$\Pi_{A} (\sigma_{C=3} (R1 \times R2))$

Output

A
1
2

Basic/ Fundamental Operators

Rename Operator:

- The Rename Operator is used to rename the output of a relation.

Symbol: rho (ρ)

Notation: $\rho_s (E)$

Where (ρ) is used to denote the rename operator.

(E) Is the result of operation which is saved with the name s.

R

A	B
P	1
Q	2

$R \times \rho_s (R)$

Rename R to S.

$R \times \rho_s (R)$

R.A	R.B	S.A	S.B
P	1	P	1
P	1	Q	2
Q	2	P	1
Q	2	Q	2

Basic/ Fundamental Operators

Union Operator (U) :

- The union operator selects all the tuples that are either in Relation R or S or in both relations R and S.
- For Union Operation to be valid, the following conditions must hold.
- Two relations R and S both have same number of attributes.
- Corresponding attribute (or column) have same domain (or type)
- The attributes of R and S must occur in the same order.

Student

Roll_no	Name
1	A
2	B
3	C
4	D

Employee

Emp_no	Name
2	B
8	G
9	H

Roll_no	Name
1	A
2	B
3	C
4	D
8	G
9	H

Basic/ Fundamental Operators

Student

Roll_no	Name
1	A
2	B
3	C
4	D

Employee

Emp_no	Name
2	B
8	G
9	H

Student U Employee

Roll_no	Name
1	A
2	B
3	C
4	D
8	G
9	H

Π Name (Student) U **Π** Name (Employee)

Basic/ Fundamental Operators

$\Pi_{\text{Name}}(\text{Student}) \cup \Pi_{\text{Name}}(\text{Employee})$

Name
A
B
C
D
G
H

Basic/ Fundamental Operators

Set Difference (-) Operator:

- Suppose R and S are two relations. The set difference operator selects all the tuples that are present in the first relation R but not in the second relation S.
- The following condition must hold for the set difference is valid.
  - Two relations R and S both have same number of attributes.
  - Corresponding attribute(or columns) have the same domain (or type). The attribute of R and S must occur in the same order .
- **Symbol:** (-)
- **Syntax : R – S**

Basic/ Fundamental Operators

Example

Student

Roll_no	Name
1	A
2	B
3	C
4	D

Employee

Emp_no	Name
2	B
8	G
9	H

(Student) – (Employee)

Roll_no	Name
1	A
3	C
4	D

Π Name (Student) - **Π** Name (Employee)

Name
A
C
D

Join Operation (⋈)

- Join is an additional/Derived operator which simplify the queries, but does not add any new power to the basic relational algebra.
- Join is a combination of a Cartesian product followed by a selection operation.

Join = Cartesian Product + selection

- A join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

Symbol: ⋈

$$A \bowtie_c B = \sigma_c (A \times B)$$

Types of Join

- **Inner join**
  - **Theta join**
  - **Equi join**
  - **Natural join**
- **Outer join**
  - **Left outer join**
  - **Right outer join**
  - **Full outer join**

Inner Join

- **Inner join**

An inner join includes only those tuples that satisfy the matching criteria, while the rest of tuples are excluded.

Theta join, equi join and natural join are called **inner join**

Theta(θ) / Conditional Join

- it combines tuples from different relations provided they satisfy the **Theta (θ)** Condition.
- It is general case of join and it is used when we want to join two or more relation based on some condition.
- The join conditions is denoted by the symbol (θ).
- It uses all kinds of comparison operator like < , >, < = , > = , = , < > .

Notation: A  **$_{\theta}$ B**

Where θ is predicate/ condition . It can be any comparison operator.

(< , >, <= , > =, =, < >)

A  **$_{\theta}$ B = σ_{θ} (A X B)**

Theta(θ) / Conditional Join

Example :

R1

A	B	C
1	8	25
2	9	30
3	10	22

S1

C	D	E
26	X	45
22	Y	43
24	Z	51

R1 X S1

A	B	R1.C	S1.C	D	E
1	8	25	26	X	45
1	8	25	22	Y	43
1	8	25	24	Z	51
2	9	30	26	X	45
2	9	30	22	Y	43
2	9	30	24	Z	51
3	10	22	26	X	45
3	10	22	22	Y	43
3	10	22	24	Z	51

Theta(θ) / Conditional Join

Example :

R1 X S1

A	B	R1.C	S1.C	D	E
1	8	25	26	X	45
3	10	22	26	X	45
3	10	22	24	Z	51

$R1 \bowtie_{R1.C < S1.C} S1 = \sigma_{R1.C < S1.C} (R1 \times S1)$

Equi Join

- When a theta join uses only equivalence (=) condition, it becomes a Equi Join.
- Equi Join is a special case of theta (or conditional) Join where condition contains equalities(=).

Notation :

$A \bowtie_{A.a1 = B.b1 \wedge \ldots\ldots\wedge A.an = B.B.bn} B$

Equi Join

Student

Sid	Name	std
101	Rohan	11
102	Mira	12

Subject

Class	Subjects
11	Maths
11	Physics
12	English
12	Chemistru

Student X Subject

Sid	Name	Std	Class	Subjects
101	Rohan	11	11	Maths
101	Rohan	11	11	Physics
101	Rohan	11	12	English
101	Rohan	11	12	Chemistry
102	Mira	12	11	Maths
102	Mira	12	11	Physics
102	Mira	12	12	English
102	Mira	12	12	Chemistry

Equi Join

Example:

Student X Subject

Sid	Name	Std	Class	Subjects
101	Rohan	11	11	Maths
101	Rohan	11	11	Physics
102	Mira	12	12	English
102	Mira	12	12	Chemistry

Student  Student.Std = Subjects.Class Subject

Natural Join

- Natural Join can only be performed if **there is at least one common attribute** (column) that exists between two relations. In addition, the attribute must have the same name and domain.
- **Notation : A**  B
- The result of the natural join is the set of all combinations of tuple in two relations A and B that are equal on their common attribute names.
- Natural Join is by default inner join because the tuples which does not satisfy the conditions of Join does not appear in the result set.

Natural Join

Example:

Courses

Cid	Course	Dept
101	Database	CS
102	Mechanics	ME
103	Electronics	EE

HOD

Dept	Head
CS	Rohan
ME	Sara
EE	Jiya

Cid	Course	Dept	Head
101	Database	CS	Rohan
102	Mechanics	ME	Sara
103	Electronics	EE	Jiya

Natural Join

$$\prod \text{Cid, Course, Dept, Head} \left(\sigma \text{Courses.Dept = HOD.Dept} \left(\text{Courses X HOD} \right) \right)$$

Outer Join

- An inner join includes only those tuples with matching attributes and the rest are discarded in the resulting relation. Therefore, we need to use outer join to include all the rest of the tuples from the participating relations in the resulting relation.
- The outer join operation is an extension of the join operation that avoids loss of information.
- Outer Join contains matching tuples that satisfy the matching condition , along with some or all tuples that do not satisfy the matching condition.
- It is based on both matched and unmatched tuple.
- It contains all rows from either one or both relations are present.
- It uses **NULL** values . NULL signifies that the value is unknown or does not exist.

Left Outer Join (R1 ⟕ R2)

- **Left Outer Join,** all the tuples from left relation R1 are included in the resulting relation . The tuples from R1 which do not satisfy Join condition , will have values as **NULL** for attributes of R2 .
- **In Short**
  - All records from left table.
  - Only matching records from right table.

Symbol : 

Notation: R1  **R2**

Left Outer Join (R1 ⟕ R2)

Example :

Courses

Cid	Course
100	Database
101	Mechanics
102	Electronics

HOD

Cid	Name
100	Rohan
102	Sara
104	Jiya

Courses ⟕ HOD

Cid	Course	Name
100	Database	Rohan
101	Mechanics	NULL
102	Electronics	Jia

Right Outer Join (R1 ⟖ R2)

- In **Right Outer Join,** all the tuples from right relation R2 are included in the resulting relation. The tuples of R2 which do not satisfy join condition will have values as **NULL** for attributes of R1.
- In short:
  - All records from right table
  - Only matching record from left table.

Symbol : ⟖

Notation : R1 ⟖ R2

Right Outer Join (R1 ⟖ R2)

Example :

Courses

Cid	Course
100	Database
101	Mechanics
102	Electronics

HOD

Cid	Name
100	Rohan
102	Sara
104	Jiya

Courses ⟖ HOD

Cid	Course	Name
100	Database	Rohan
102	Electronics	Sara
104	NULL	Jia

Full Outer Join (R1 R2)

- In **Full Outer Join,** all the tuples from both left relation R1 and right relation R2 are included in the resulting relation. The tuples of both relations R1 and R2 which do not satisfy Join condition, their respective unmatched attributes are made **NULL .**
- In short:
  - All records from all tables.

Symbol:

Notation : R1 R2

Full Outer Join (R1 ⟗ R2)

Example :

Courses

Cid	Course
100	Database
101	Mechanics
102	Electronics

HOD

Cid	Name
100	Rohan
102	Sara
104	Jiya

Courses ⟗ HOD

Cid	Course	Name
100	Database	Rohan
101	Mechanics	NULL
102	Electronics	Sara
104	NULL	Jiya

Division Operator

- ✓Division Operation is represented by "division"(÷ or /) operator and is used in queries that involve keywords **"every"**, **"all"**, etc.
  Notation : R(X,Y)/S(Y)
  Here,
  R is the first relation from which data is retrieved.
  S is the second relation that will help to retrieve the data.
  X and Y are the attributes/columns present in relation. We can have multiple attributes in relation, but keep in mind that attributes of S must be a proper subset of attributes of R.
  For each corresponding value of Y, the above notation will return us the value of X from tuple<X,Y> which exists **everywhere**.

Example

Enrolled

STUDENT_ID	COURSE_ID
Student_1	DBMS
Student_2	DBMS
Student_1	OS
Student_3	OS

Course

COURSE_ID
DBMS
OS

ENROLLED(STUDENT_ID, COURSE_ID)/COURSE(COURSE_ID)

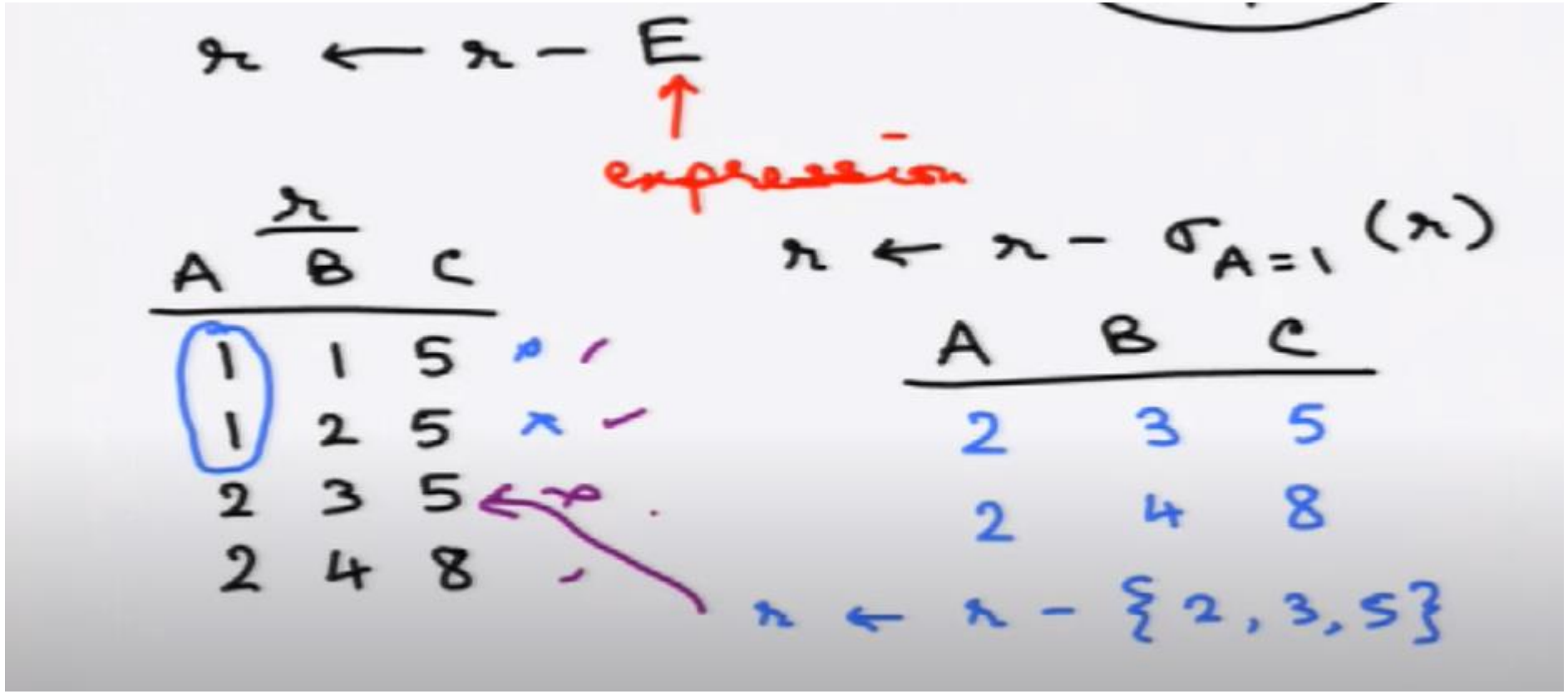
STUDENT_ID
Student_1

the query is to return the STUDENT_ID of students who are enrolled in **every** course

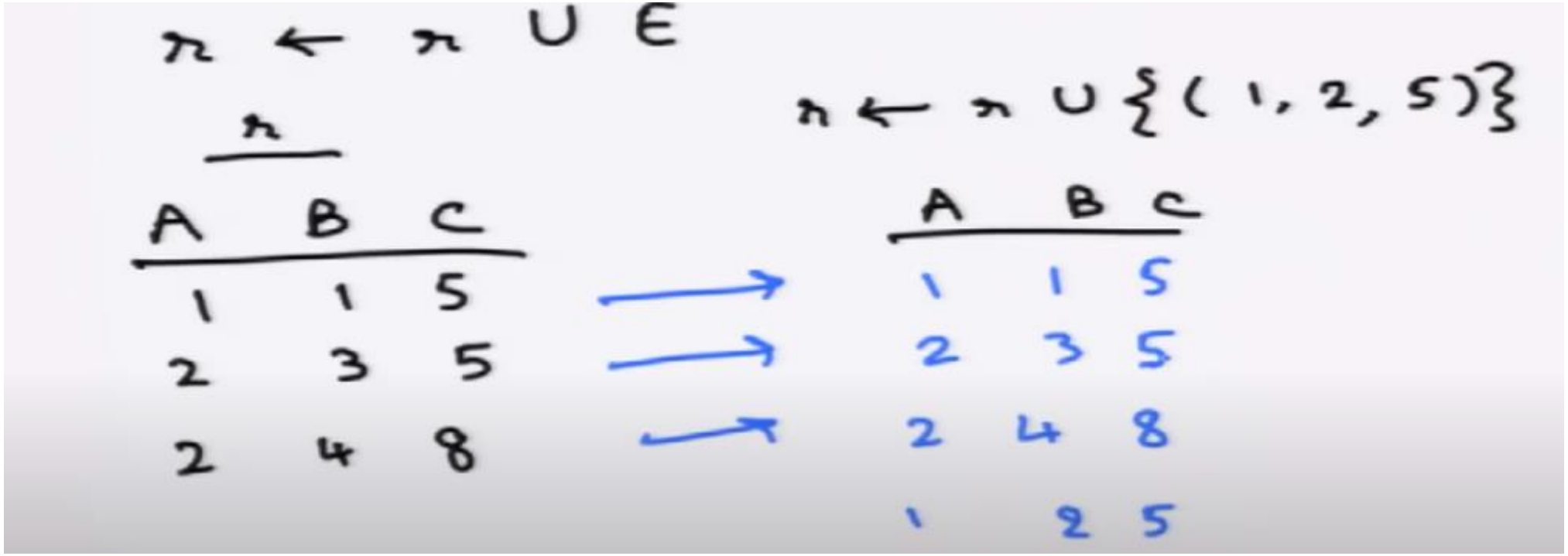
Database Modification

- Deletion
- Insertion
- Updation

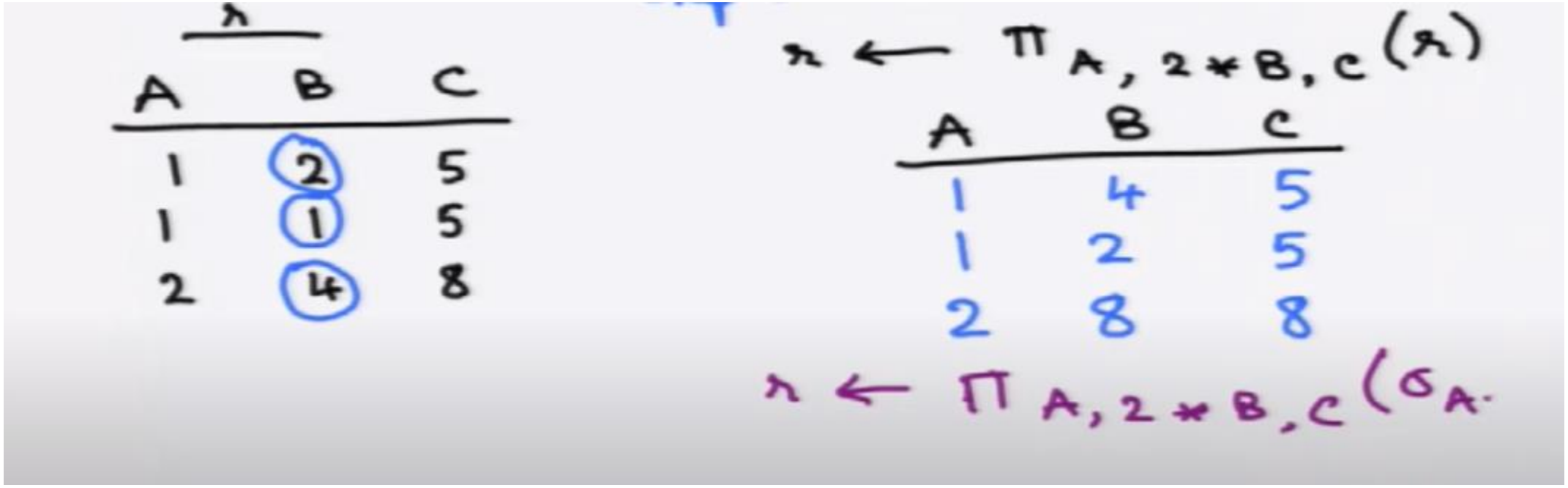
Deletion



Insertion



Updation



Update, delete and insert

- https://www.youtube.com/watch?v=rfH9Q4gS9BQ
- https://www.scaler.com/topics/dbms/relational-algebra-in-dbms/
- https://www.linkedin.com/pulse/02-dml-relational-algebra-build-your-own-dbms-from-first-el-deen/