Unit 2 – Relational Model

Prof. Sumit P. Makwana

Table of Contents

Relational Algebra

- Relational Algebra is procedural query language, which takes Relation as input and generate relation as output.
- Relational algebra mainly provides theoretical foundation for relational databases and SQL.
- In relational algebra input is a relation (table) and output is also a relation (table) (temporary table holding the data asked by the user)

Operations in Relational Algebra

- Unary Relational Operation
  - SELECT (Symbol σ) (Sigma)
  - PROJECT (Symbol ∏) (Pi)

- Binary Relational Operation
  - Operations from set theory
    - UNION (Symbol ∪)
    - INTERSECTION (Symbol ∩)
    - DIFFERENCE (Symbol -)
    - CARTESIAN PRODUCT (Symbol X)
  - JOIN
  - Inner JOIN
  - Left Outer JOIN
  - Right Outer JOIN
  - Full Outer JOIN

SELECT Operation

Symbol : σ (SIGMA)

Notation: $\sigma_{condition}$ (Relation_Name)

Operation: Display particular tuples from a relation that satisfy a given condition (predicate)

Operators: =, <>, <, >, <=, >=, ^ (AND), v (OR)

Example 1:

Select all account whose balance is greater than 100000 from relation SavingAaccount

$$\sigma_{balance > 100000}(SavingAccount)$$

Example 2:

Below is the relation named 'r'. From relation r select the records where a = b and d > 5. When we use the below Select operation to find the below output relation is generated.

Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$$\sigma_{a=b \wedge D>5}(r)$$

Output Relation

A	B	C	D
α	α	1	7
β	β	23	10

Exercise

Student

Enroll.no	Name	Branch	Age	Birthdate	City
S1	Dev	IT	18	01-05-2002	Rajkot
S2	Karan	CE	20	19-03-2000	Mumbai
S3	Jatin	ME	20	04-02-2000	Baroda
S4	Meet	IT	17	16-08-1999	Delhi
S5	Niraj	EC	19	20-02-2001	Patna

Above is the relation Student. Find below using Select operation

Questions

1. Display the detail of students belongs to “Rajkot” city.
2. Display the details students whose name is Karan
3. Display the details of students who is studying in EC branch
4. Display the details of students who are living in Mumbai and studying in CE branch

5. Display the details of students who are living in Delhi and Patna
6. Display the details of students whose enrolment no is S3 and S5

Answers

1. $\sigma_{city='Rajkot'}(Student)$
2. $\sigma_{name='Karan'}(Student)$
3. $\sigma_{branch='EC'}(Student)$
4. $\sigma_{city='Mumbai' \vee branch='CE'}(Student)$
5. $\sigma_{city='Delhi' \vee city='Patna'}(Student)$
6. $\sigma_{enroll.no='S3' \vee enroll.no='S5'}(Student)$

PROJECT Operation

Symbol: Π (Pi)

Notation: $\pi_{attribute\_list}(Relation\_name)$

- Operation: Selects specified attributes of a relation.
- Project operation selects certain columns from a table while discarding others.
- It removes any duplicate tuples (records) from the result.
- The result of the project operation has only the attributes specified in the attribute list and in the same order as they appear in list.

Example 1

List out all account number and name from SavingAccount relation

$\pi_{acc\_no,\ name}(SavingAccount)$

Example 2

Relation r

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

Output relation

A	C
α	1
α	1
β	1
β	2

→

A	C
α	1
β	1
β	2

Exercise

Student

Enrollno	Name	Branch	Age	Birthdate	City
S1	Dev	IT	18	01-05-2002	Rajkot
S2	Karan	CE	20	19-03-2000	Mumbai
S3	Jatin	ME	20	04-02-2000	Baroda
S4	Meet	IT	17	16-08-1999	Delhi
S5	Niraj	EC	19	20-02-2001	Patna

Above is the relation Student. Find below using Project operation

Questions

1. Display the Enrollno, Name and city of all students.
2. Display Enrollno, branch, name of students
3. Display branch, name, city of students
4. Display name, Enrollno, branch of students

Answers

1. $\pi_{\text{enrollno, name}}$ (Student)
2. $\pi_{\text{enrollno, branch, name}}$ (Student)
3. $\pi_{\text{branch, name, city}}$ (Student)
4. $\pi_{\text{name, enrollno, branch}}$ (Student)

Combination of SELECT and PROJECT Operation

We can also combine Select and Project operation as seen in below example.

Student

Enrollno	Name	Branch	Age	Birthdate	City
S1	Dev	IT	18	01-05-2002	Rajkot
S2	Karan	CE	20	19-03-2000	Mumbai
S3	Jatin	ME	20	04-02-2000	Baroda
S4	Meet	IT	17	16-08-1999	Delhi
S5	Niraj	EC	19	20-02-2001	Patna

Display the Enrollno,Name and city of "IT" branch students.

$$\pi_{\text{enrollno, name, city}} (\sigma_{\text{branch='IT'}} (\text{Student}))$$

Output Relation

Output Relation

Enrollno	Name	City
S1	Dev	Rajkot
S4	Meet	Delhi

UNION Operation

Symbol ∪
Notation: Relation1_name ∪ Relation2_name

- **Operation:** Operation of Union in relational algebra is same as set theory union operation. It combines the tuples (records) of the both input relation.
- **Requirement:** Union must be taken between compatible relations
- Relation R and S are compatible, if

- Both have same number of attributes
- Domain of the attribute of R and S are similar

Example 1

Relation r

A	B
α	1
α	2
β	1

Relation s

A	B
α	2
β	3

r ∪ s

Output

A	B
α	1
α	2
β	1
β	3

In above example we have two relation named 'r' and 's'. Output of R ∪ S will display all the records of relation r and relation after combining them. If there are some duplicate records it will remove the duplicate records

Example 2

Employee

Name	City	Salary	Department
Ravi	Rajkot	50000	CE
Madhu	Jamnagar	38000	CE
Ankita	Ahmedabad	40000	IT
Tejas	Bhuj	45000	ME

Student

Enrol_No	Name	Class	Department
1001	Bhargav	DC1	CE
1002	Bhavik	DC2	IT
1003	Madhu	DC1	ME

Above are two relation employee and relation that stores the detail of employee and student respectively.

List all the students and faculties with their department.

Here Employee relation and Student relation are not compatible relations. But the resultant relation of projection operator contains similar attribute with similar domain. So, it is possible to apply union operation after applying such project operation even though the original relations are not compatible.

$\pi_{name, department}(Employee) \cup \pi_{name, department}(Student)$

Output

Name	Department
Ravi	CE
Madhu	CE
Ankita	IT
Tejas	ME
Bhargav	CE
Bhavik	IT
Madhu	ME

INTERSECTION Operation

Symbol ∩
Notation: Relation1_name ∩ Relation2_name

- **Operation:** Selects the tuples (records) which are common in both the input relations.
- **Requirement:** Union must be taken between compatible relations
- Relation R and S are compatible, if
  - Both have same number of attributes
  - Domain of the attribute of R and S are similar

Example 1

Relations r

A	B
α	1
α	2
β	1

Relations s

A	B
α	2
β	3

r ∩ s

Output

A	B
α	2

In above example we have two relation named 'r' and 's'. Output of **R ∩ S** will display only those records which are common in both the relation.

Example 2

Employee

Name	City	Salary	Department
Ravi	Rajkot	50000	CE
Madhu	Jamnagar	38000	CE
Ankita	Ahmedabad	40000	IT
Tejas	Bhuj	45000	ME

Student

Enrol_No	Name	Class	Department
1001	Bhargav	DC1	CE
1002	Bhavik	DC2	IT
1003	Madhu	DC1	CE

Above are two relation employee and relation that stores the detail of employee and student respectively.

List all the employee who are also students.

Here Employee relation and Student relation are not compatible relations. But the resultant relation of projection operator contains similar attribute with similar domain. So, it is possible to apply intersection operation after applying such project operation even though the original relations are not compatible.

$$\pi_{\text{name, department}}(\text{Employee}) \cap \pi_{\text{name, department}}(\text{Student})$$

Output

Name	Department
Madhu	CE

SET DIFFERENCE Operation

- **Symbol:** (minus sign)
- **Notation:** Relation1_name Relation2_name
- **Operation:** Returns all the records from relation1 (left relation) after removing the common records of relation1 and relation2.

Example

Employee

Name	City	Salary	Department
Ravi	Rajkot	50000	CE
Madhu	Jamnagar	38000	CE
Ankita	Ahmedabad	40000	IT
Tejas	Bhuj	45000	ME

Student

Enrol_No	Name	Class	Department
1001	Bhargav	DC1	CE
1002	Bhavik	DC2	IT
1003	Madhu	DC1	CE

Above are two relation employee and relation that stores the detail of employee and student respectively.

List all the employee who are not students.

Here Employee relation and Student relation are not compatible relations. But the resultant relation of projection operator contains similar attribute with similar domain. So, it is possible to apply set difference operation after applying such project operation even though the original relations are not compatible.

$$\pi_{name}(Employee) - \pi_{name}(Student)$$

Output

Name
Ravi
Ankita
Tejas

CROSS PRODUCT Operation

CROSS PRODUCT Operation is also known as CARTESIAN PRODUCT Operation

Symbol: X (Cross)

Notation: Relation1_name X Relation2_name

Operation: Combines information of two relations, it is also known as Cross-Product operation and similar to mathematical Cartesian Product Operation

Result: for Relation1 and Relation2 if they have n1 and n2 attributes respectively then resultant relation will have n1+n2 attributes, combining attributes from both the input relations

If both relations have same name of attributes, then they are distinguished using following notation.

```
Relation_Name.Attribute_Name
```

i.e., Relation1.n1 or Relation2.n1

If Relation1 has t1 tuples and Relation2 has t2 tuples then the resultant relation will have t1 * t2 tuples, combining each possible pair of tuples from both the relations.

In short, we can say,

```
Attributes of Resultant Relation = Attributes of R1
+ Attributes of R2

Tuples of Resultant Relation = Tuples of R1 *
Tuples of R2
```

Example

Student

RollNo	Name	Branch
101	Raj	CE
102	Meet	ME

Result

RollNo	SPI
101	8
103	9

Above are two relations. Student relation have three attributes roll no, name and branch and two records. Result relation have two attribute roll number and SPI and two records.

When we cross product these two relations

The resultant relation will have total five attribute (Attributes of Student + Attributes of Result)

And will have total 4 records. (Records of Student x Records of Result) (All records of student relation are matched with every record of the result relation.)

Student

RollNo	Name	Branch
101	Raj	CE
102	Meet	ME

Result

RollNo	SPI
101	8
103	9

Output Relation

Student × Result

Student.RollNo	Name	Branch	Result.RollNo	SPI
101	Raj	CE	101	8
101	Raj	CE	103	9
102	Meet	ME	101	8
102	Meet	ME	103	9

In above example both relation have same attribute RollNo so it is distinguished by notation Relation_name.Attribute_Name (i.e. Student.RollNo and Result.RollNo)

Join

In Cross Product all records of left table are matched with every record of the right table. In cross product many records are inconsistent or the data is not correct.

To get the consistent and correct data we can use join.

There are four types of join

- Natural Join (Inner Join)

- Left Outer Join
- Right Outer Join
- Full Outer Join

Natural Join (Inner Join)

Symbol: ⋈
Notation: Relation_1 ⋈ Relation_2

Operation: Natural join will retrieve consistent data from multiple relations. It combines records from different relations that satisfy a given condition. To perform a natural join there must be one common attribute (column) between two relations.

Step	Operation
Step 1	It performs Cartesian Product
Step 2	Then it deletes inconsistent tuples
Step 3	Then it removes an attribute from duplicate attributes

Example 1

Input Relations

Student

Roll_no	Name	Branch
1001	Bhim	CE
1002	Nakul	CE
1003	Arjun	IT

Result

Roll_no	Subject	Mark
1001	Maths	70
1001	Physics	80
1003	Biology	90

When we perform natural Join

Student ⋈ Result

Step 1. Perform Cross Product Operation

Student.Roll_no	Name	Branch	Result.Roll_no	Subject	Mark
1001	Bhim	CE	1001	Maths	70
1001	Bhim	CE	1001	Physics	80
1001	Bhim	CE	1003	Biology	90
1002	Nakul	CE	1001	Maths	70
1002	Nakul	CE	1001	Physics	80
1002	Nakul	CE	1003	Biology	90
1003	Arjun	IT	1001	Maths	70
1003	Arjun	IT	1001	Physics	80
1003	Arjun	IT	1003	Biology	90

Step 2. Remove Inconsistent Records

Student.Roll_no	Name	Branch	Result.Roll_no	Subject	Mark
1001	Bhim	CE	1001	Maths	70
1001	Bhim	CE	1001	Physics	80
1003	Arjun	IT	1003	Biology	90

Step 3. Remove Duplicate Attribute

Roll_no	Name	Branch	Subject	Mark
1001	Bhim	CE	Maths	70
1001	Bhim	CE	Physics	80
1003	Arjun	IT	Biology	90

Example 2

Input Relations (Course & Head)

CID	Course	Dept
CS01	Database	CS
ME01	Mechanics	ME
EE01	Electronics	EE
PH01	Physics	ME

Dept	Head
CS	Alex
ME	Maya
EE	Mira

When we perform natural Join

Student ⋈ Result

Step 1. Perform Cross Product Operation

CID	Course	Course.Dept	Head.Dept	Head
CS01	Database	CS	CS	Alex
CS01	Database	CS	ME	Maya
CS01	Database	CS	EE	Mira
ME01	Mechanics	ME	CS	Alex
ME01	Mechanics	ME	ME	Maya
ME01	Mechanics	ME	EE	Mira
EE01	Electronics	EE	CS	Alex
EE01	Electronics	EE	ME	Maya
EE01	Electronics	EE	EE	Mira
PH01	Physics	ME	CS	Alex
PH01	Physics	ME	ME	Maya
PH01	Physics	ME	EE	Mira

Step 2. Remove Inconsistent Records

CID	Course	Course.Dept	Head.Dept	Head
CS01	Database	CS	CS	Alex
ME01	Mechanics	ME	ME	Maya
EE01	Electronics	EE	EE	Mira
PH01	Physics	ME	ME	Maya

Step 3. Remove Duplicate Attribute

CID	Course	Course	Head
CS01	Database	CS	Alex
ME01	Mechanics	ME	Maya
EE01	Electronics	EE	Mira
PH01	Physics	ME	Maya

Left Outer Join

In inner join, only the data which have matching records on either side is displayed. If some records do not have matching records the it will be omitted.

To view the records even if there are no matching records on other table we can use left or right outer join.

Symbol: ⟕

Notation: Relation1 ⟕ Realtion2

Use: Combine multiple relation using some common attribute and also display all the tuples of the left relation even through there is no matching tuple in the right relation.

If there are tuples in Left relation without any matching tuple in the Right relation, then the Right relation attributes of the resulting relation are treated as NULL.

Example

Input Relations (Course & Head)

CID	Course	Dept
CS01	Database	CS
ME01	Mechanics	ME
MA01	Electronics	EE
PH01	Physics	ME

Dept	Head
CS	Alex
ME	Maya

When we perform left outer join on above relations (tables)

Course ⟕ Head

Output of Left Outer Join

CID	Course	Course.Dept	Head
CS01	Database	CS	Alex
ME01	Mechanics	ME	Maya
EE01	Electronics	EE	NULL
PH01	Physics	ME	Maya

Right Outer Join

Symbol: ⟖

Notation: Relation1 ⟖ Realtion2

Use: Combine multiple relation using some common attribute and also display all the tuples of the right relation even through there is no matching tuple in the left relation.

If there are tuples in right relation without any matching tuple in the left relation, then the left relation attributes of the resulting relation are treated as NULL.

Example

Input Relations (Course & Head)

CID	Course	Dept
CS01	Database	CS
ME01	Mechanics	ME
MA01	Electronics	EE

Dept	Head
CS	Alex
ME	Maya
Auto	Michael
Civil	Mike

When we perform right outer join on above relations (tables)

Course ⟖ Head

Output of Right Outer Join

CID	Course	Course.Dept	Head
CS01	Database	CS	Alex
ME01	Mechanics	ME	Maya
NULL	NULL	Auto	Michael
NULL	NULL	Civil	Mike

Full Outer Join

Symbol: ⟗

Notation: Relation1 ⟗ Realtion2

Use: Display all the records from both relations even if there are no matching records on the either side.

If no matching records found on either side, then NULL value is used for such records.

Basically, full outer join = left outer join + right outer join.

Example

Input Relations (Square & Cube)			
Number	**Square**	**Number**	**Cube**
2	4	1	1
3	9	3	27
4	16	5	125
5	25	10	1000

Output of Full Outer join

Square ⟗ Cube

Number	Square	Cube
1	NULL	1
2	4	NULL
3	9	27
4	16	NULL
5	25	125
10	NULL	1000

Relational Calculus

Like relational algebra relational calculus is an abstract database language

Relational calculus and relational algebra can achieve same results.

In relational calculus queries are described as {|} with desired properties.

There are two flavors of relational calculus

- Tuple
- Domain

Tuple Relational Calculus

The tuple relational calculus is specified to select the tuples in a relation.

```
{t | P (t)} or {t | condition (t)}
```

this is also known as expression of relational calculus

Where t is the resulting tuples, P(t) is the condition used to fetch t.

Example 1

```
{t | EMPLOYEE (t) and t.SALARY>10000}
```

Reads as “set of all t such that t is an element of the relation EMPLOYEE and the SALARY attribute has value greater than 10000”

Implies that it selects the tuples from EMPLOYEE relation such that resulting employee tuples will have salary greater than 10000. It is example of selecting a range of values.

Example 2

```
{t | EMPLOYEE (t) AND t.DEPT_ID = 10}
```

Above expression selects all the tuples of employee name who work for Department 10.

The variable which is used in the condition is called tuple variable.

In above examples t.SALARY and t.DEPT_ID are tuple variables.

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 Λ (AND), V (OR) and ! (not).

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

Where a1, a2 are attributes
P stands for formula (condition) built by inner attributes

Example

```
{< name, age > | < name, age > ϵ Student Λ age > 17}
```

Again, the above query will return the names and ages of the students in the table Student who are older than 17.

Aggregate Functions

Symbol: G
Notation: $G_{\text{function-name(column)}}$ (Relation)

Use: It takes collection of values as input and returns a single value as output.

E.g., Sum, max, min, avg, count

Sum

It finds the sum of values in particular column.

Example

Employee			
ID	**Name**	**Salary**	**Post**
1	Meet	15000	Assistant
2	Jay	20000	Manager
3	Ajay	50000	Assistant
4	Vijay	60000	Sr. Manager

Find the sum of the salary of all employees

$$G_{sum(salary)}\ (Employee)$$

Output

Salary
145000

Max

It is used to find the maximum value in particular column

Example

Student			
ID	**Name**	**CPI**	**SPI**
1	Meet	7	7.5
2	Jay	7.5	7.6
3	Ajay	9	9.5
4	Vijay	5.5	5.2
5	Arjun	6.6	7.1
6	Lakshman	8.8	8.5

Find the Maximum CPI and SPI from Student table.

$$G_{max(CPI),\ max(SPI)}\ (Student)$$

Output

CPI	CPI
9	9.5

Min

It is used to find the minimum value in particular column

Example

Student			
ID	**Name**	**CPI**	**SPI**
1	Meet	7	7.5
2	Jay	7.5	7.6
3	Ajay	9	9.5
4	Vijay	5.5	5.2
5	Arjun	6.6	7.1
6	Lakshman	8.8	8.5

Find the Minimum CPI and SPI from Student table.

$$G_{min(CPI),\ min(SPI)}\ (Student)$$

Output

CPI	SPI
5.5	5.2

Avg

It is used to find the average value of particular column

Example

Student			
ID	**Name**	**CPI**	**SPI**
1	Meet	7	7.5
2	Jay	7.5	7.6
3	Ajay	9	9.5
4	Vijay	5.5	5.2
5	Arjun	6.6	7.1
6	Lakshman	8.8	8.5

Find the Average CPI and SPI from Student table.

$$G_{avg(CPI),\ avg(SPI)}\ (Student)$$

Output

CPI	SPI
7.4	7.56

Count

It is used to count the rows or number of values in particular column.

Note it only counts the values, if the column has NULL value, it is not counted.

Example

Student			
ID	**Name**	**CPI**	**SPI**
1	Meet	7	7.5
2	Jay	7.5	7.6
3	Ajay	9	9.5
4	Vijay	5.5	5.2
5	Arjun	6.6	7.1
6	Lakshman	8.8	8.5

Find the number of Students

$G_{count(*)}$ (Student)
or
$G_{count(ID)}$ (Student)

Output

count(ID)
6

Here count (*) will always give the number of rows in a table.

Count (column_name) will give the number of values in that particular row. If column has NULL values its value will be less than count (*) as NULL values are not counted.