Relational Algebra

What is Relational Algebra?

- A formal way to express a query in relational model
  - A query consists of relational expressions describing the sequence of operators applied to obtain the query result

- Example :

$\pi_{LName, Address}(\sigma_{DNo=4 \text{ OR } Salary>100000}(\text{EMPLOYEE}))$

  - π and σ are the operators

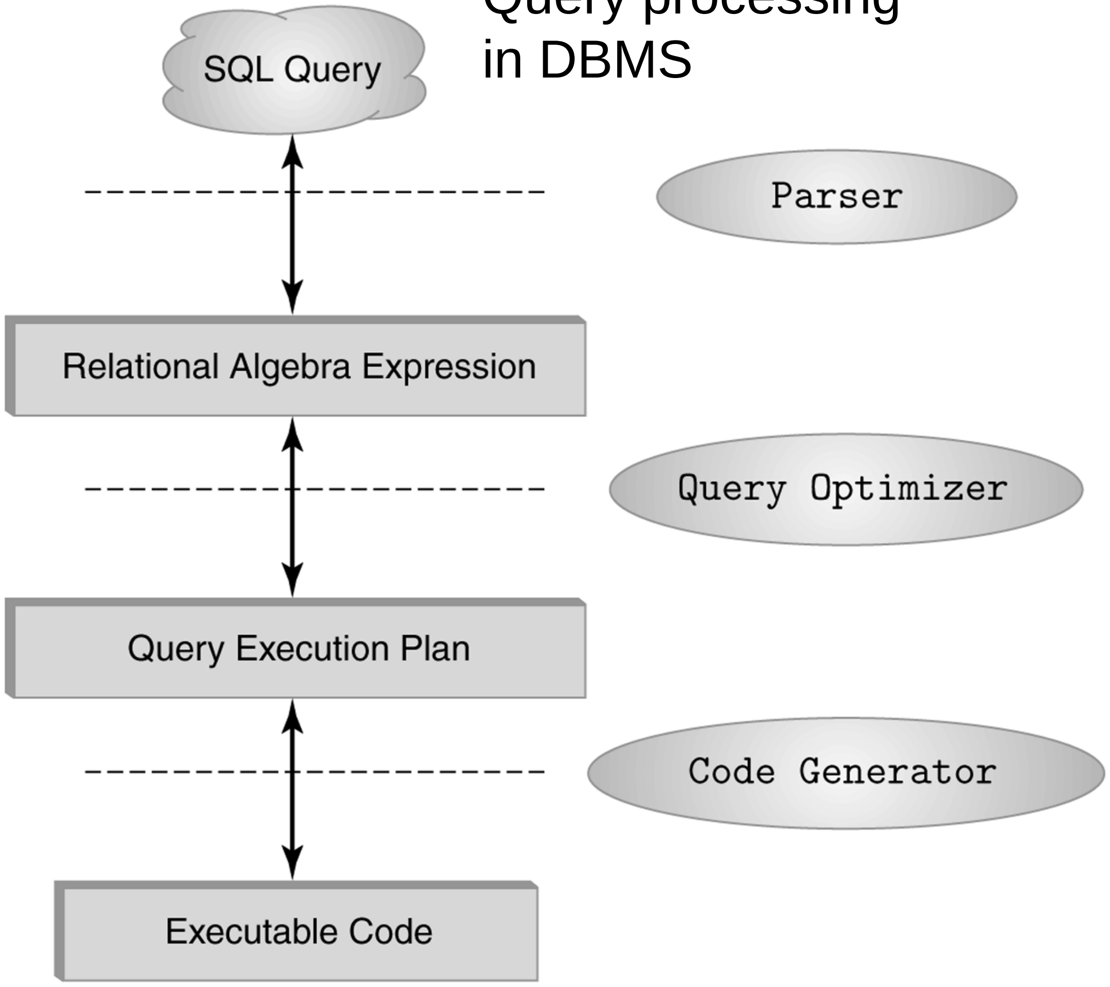
Why study Relational Algebra?

SELECT A.Name, B.Grade
FROM A, B
WHERE A.Id = B.Id

$$\downarrow$$

$\pi_{\text{Name, Grade}}(\sigma_{\text{Id,Name}}(A \times B))$

Query processing in DBMS



Relational Algebra Operator

1. Select (σ)
2. Project (∏)
3. Union (∪)
4. Set Difference (-)
5. Cartesian product (X)
6. Rename (ρ)

Select Operator (σ)

$\sigma_{condition}(R)$ **(NOT THE SAME AS SELECT in SQL)**

- This operation is used to select tuples (rows) from a table (relation) that specifies a given logic, which is called as a predicate. The predicate is a user defined condition to select rows of user's choice.
- you can think of it as a **where clause in SQL**, which is used for the same purpose.
  - contains only tuples (rows) in *R* that satisfy the given condition

Example: Student

Id	*Name*	*Address*	*Status*
1123	John	123 Main	Junior
1234	Lee	123 Main	Senior
5556	Mary	7 Lake Dr	Freshman
9876	Bart	5 Pine St	Junior

$\sigma_{Status=\text{'Junior'}}(\text{Student})$

Id	*Name*	*Address*	*Status*
1123	John	123 Main	Junior
9876	Bart	5 Pine St	Junior

Selection Condition - Examples

Student

Id	*Name*	*Address*	*Status*
1123	John	123 Main	Junior
1234	Lee	123 Main	Senior
5556	Mary	7 Lake Dr	Freshman
9876	Bart	5 Pine St	Junior

Selectivity

Fraction of tuples selected by a selection condition

- $\sigma_{\textbf{Id>3000 OR Status='Freshman'}}$ (Student) selectivity = 2/4
- $\sigma_{\textbf{Id>3000 AND Id <3999}}$ (Student) selectivity = 0/4
- $\sigma_{\textbf{NOT(Status='Senior')}}$ (Student) selectivity = 3/4
- $\sigma_{\textbf{Status}\neq\textbf{'Senior'}}$ (Student) selectivity = 3/4

Project Operator (∏)

- Project operator is denoted by ∏ symbol and it is used to select desired columns (or attributes) from a table (or relation).

- Project operator in relational algebra is similar to the **Select statement in SQL**.

Project Operator (∏)

$\pi_{\textbf{attribute list}}\textbf{(R)}$

- Produces a relation containing a <u>subset of columns</u> of its input relation.
- If the user is interested in selecting the values of a few attributes, rather than selection all attributes of the Table (Relation), then one should go for PROJECT Operation.
- **<u>Example</u>: Student**

Id	*Name*	*Address*	*Status*
1123	John	123 Main	Junior
1234	Lee	123 Main	Freshman
5556	Mary	7 Lake Dr	Senior
9876	Bart	5 Pine St	Junior

$\pi_{\text{Name,Status}}(\text{Student})$

Name	*Status*
John	Junior
Lee	Freshman
Mary	Senior
Bart	Junior

Project Operator

- Resulting relation has no duplicates; therefore it can have fewer tuples than the input relation

- Example:

Student

Id	*Name*	*Address*	*Status*
1123	John	123 Main	Junior
1234	Lee	123 Main	Freshman
5556	Mary	7 Lake Dr	Senior
9876	Bart	5 Pine St	Junior

$\pi_{Address}$(Student)

Address
123 Main
7 Lake Dr
5 Pine St

Relational Algebra Expressions

- Find the Ids and names of junior undergraduate students

Student

Id	*Name*	*Address*	*Status*
1123	John	123 Main	Junior
1234	Lee	123 Main	Freshman
5556	Mary	7 Lake Dr	Senior
9876	Bart	5 Pine St	Junior

Id	*Name*
1123	John
9876	Bart

Result

$$\pi_{Id,\ Name}\ (\sigma_{Status='\text{Junior}'}\ (\text{Student})\ )$$

Relational Algebra Expressions

- We may want to apply several relational algebra operations one after the other
  - We can write the operations as a single **relational algebra expression** by nesting the operations, or
  - We can apply one operation at a time and create **intermediate result relations**.

  - In the latter case, we must give names to the relations that hold the intermediate results.

Single versus Sequence of operations

Employee

Fname	Lname	<u>SSN</u>	BDate	Salary	Address	DNo

- Query: Find the first name, last name, and salary of all employees who work in department number 5

Single versus Sequence of operations

Employee

Fname	Lname	<u>SSN</u>	BDate	Salary	Address	DNo

- Query: Find the first name, last name, and salary of all employees who work in department number 5

- We can write a *single relational algebra expression:*
  - $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO=5}}(\text{EMPLOYEE}))$

RENAME Operator

- The RENAME operator is denoted by ρ

- We may *rename* the attributes of a relation or the relation name or both
  - $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$
    - Change the relation name from *R* to *S, and*
    - Change column (attribute) names to B1, B2, …..Bn

  - $\rho_{S}(R)$:
    - Change the *relation name* only to S

  - $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$:
    - Change the *column (attribute) names* only to B1, B2, …..Bn

Example

- Change relation name to Emp and attributes to First_name, Last_name, and Salary

$\rho_{\text{Emp(First\_name, Last\_name, Salary)}} (\pi_{\text{FNAME,LNAME,SALARY}} \text{(EMPLOYEE)})$

or

$\text{Emp(First\_name, Last\_name, Salary)} \leftarrow \pi_{\text{FNAME,LNAME,SALARY}}\text{(EMPLOYEE)}$

Example

EMPLOYEE (Ssn, Salary, Deduction, Years_service)

A report may be required to show as follow,

- Net Salary = Salary – Deduction,
- Bonus = 2000 * Years_service, and
- Tax = 0.25 * Salary

Then a generalized projection combined with <u>renaming</u> may be used as follows:

$$\text{REPORT} \leftarrow \rho_{(\text{Ssn, Net\_salary, Bonus, Tax})}(\pi_{\text{Ssn, Salary – Deduction, } 2000 * \text{Years\_service,}}$$

$$* \text{ Salary}(\text{EMPLOYEE})).$$

Set Operators

- A relation is a set of tuples; therefore set operations are applicable:
  - **<u>Intersection</u>:** $\cap$
  - **<u>Union</u>:** $\cup$
  - **<u>Set difference</u>** (Minus): $-$

- <u>*Set operators are binary operators*</u>
  - Operator: Relation $\times$ Relation $\rightarrow$ Relation

- Result of set operation is a relation that has the same schema as the combining relations

Examples: Set Operations

X

A	B
x1	x2
x3	x4

Y

A	B
x1	x2
x5	x6

X ∩ Y

A	B
x1	x2

X ∪ Y

A	B
x1	x2
x3	x4
x5	x6

X – Y

A	B
x3	x4

Example

(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

STUDENT
∪
INSTRUCTOR

STUDENT – INSTRUCTOR

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

STUDENT
∩
INSTRUCTOR

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

INSTRUCTOR – STUDENT

UNION OPERATOR (U)

- Union operator is denoted by ∪ symbol and it is used to select all the rows (tuples) from two tables (relations).
- Lets say we have two relations R1 and R2 both have same columns and we want to select all the tuples(rows) from these relations then we can apply the union operator on these relations.

- **Note:** The rows (tuples) that are present in both the tables will only appear once in the union set. In short you can say that there are <u>no duplicates</u> present after the union operation.

Union Operator (∪): Compatibility

- Two relations are *union compatible* if
  - Both have same number of columns
  - Names of attributes are the same in both
  - Attributes with the same name in both relations have the same domain

Tables:

Student (*SSN*, *Name*, *Address*, *Status*)
Professor (*Id*, *Name*, *Office*, *Phone*)

are **not** union compatible.

But,

π_{Name} (Student) and π_{Name} (Professor)

are union compatible

Exercise

Relations:

- Employee (*SSN*, *Name*, *Address*)
- Professor (*SSN*, *Office*, *Phone*)
- Student (*SSN*, *Status*)

- Find the SSN of student and employees.
- Find the SSN of employees who are not professors
- Find the SSN of employees who are neither professors nor students.

Exercise

Relations:

- Employee (<u>*SSN*</u>, *Name*, *Address*)
- Professor (<u>*SSN*</u>, *Office*, *Phone*)
- Student (<u>*SSN*</u>, *Status*)

- Find the SSN of student and employees.

  $\pi_{SSN}(\text{Employee}) \cap \pi_{SSN}(\text{Student})$

- Find the SSN of employees who are not professors

- Find the SSN of employees who are neither professors nor students

Exercise

- Relations:
  - Employee (<u>*SSN*</u>, *Name*, *Address*)
  - Professor (<u>*SSN*</u>, *Office*, *Phone*)
  - Student (<u>*SSN*</u>, *Status*)

- Find the SSN of student and employees.

  $\pi_{SSN}(\text{Employee}) \cap \pi_{SSN}(\text{Student})$

- Find the SSN of employees who are not professors

  $\pi_{SSN}(\text{Employee}) - \pi_{SSN}(\text{Professor})$

- Find the SSN of employees who are neither professors nor students

Exercise

- Relations:
  - Employee (<u>*SSN*</u>, *Name*, *Address*)
  - Professor (<u>*SSN*</u>, *Office*, *Phone*)
  - Student (<u>*SSN*</u>, *Status*)

- Find the SSN of student and employees.

  $\pi_{SSN}(\text{Employee}) \cap \pi_{SSN}(\text{Student})$

- Find the SSN of employees who are not professors

  $\pi_{SSN}(\text{Employee}) - \pi_{SSN}(\text{Professor})$

- Find the SSN of employees who are neither professors nor students

  $\pi_{SSN}(\text{Employee}) - (\pi_{SSN}(\text{Student}) \cup \pi_{SSN}(\text{Professor}))$

Cartesian Product

A	*B*
x1	x2
x3	x4

R

C	*D*
y1	y2
y3	y4

S

$\longrightarrow$

A	*B*	*C*	*D*
x1	x2	y1	y2
x1	x2	y3	y4
x3	x4	y1	y2
x3	x4	y3	y4

$R \times S$

- R × S is expensive to compute:
  - Number of columns = degree(R) + degree(S)
  - Number of rows = number of rows (R) × number of rows (S)

Example

Broker (*<u>BrokerId</u>*, *BrokerName*)

Client (*<u>ClientId</u>*, *ClientName*)

<u>**BrokerID**</u>	**BrokerName**
1	Merrill Lynch
2	Morgan Stanley
3	Salomon Smith Barney

<u>**Client Id**</u>	**Client Name**
A11	Bill Gates
B11	Steve Jobs

- List all the possible Broker-Client pairs

Broker × Client

BrokerId	**BrokerName**	**ClientId**	**ClientName**
1	Merrill Lynch	A11	Bill Gates
1	Merrill Lynch	B11	Steve Jobs
2	Morgan Stanley	A11	Bill Gates
2	Morgan Stanley	B11	Steve Jobs
3	Salomon Smith Barney	A11	Bill Gates
3	Salomon Smith Barney	B11	Steve Jobs

Summary (Relational Algebra Operators)

- **Unary operators**
  - SELECT $\sigma_{condition}(R)$
  - PROJECT $\pi_{Attribute\text{-}List}(R)$
  - RENAME $\rho_{S(A1,A2,\ldots Ak)}(R)$
- **Set operators**
  - $R \cup S$
  - $R \cap S$
  - $R - S$ or $S - R$
- **Cartesian product**
  - $R \times S$

DIVISION OPERATOR

- Division operator A÷B can be applied if and only if:
- Attributes of B is proper subset of Attributes of A.
- The relation returned by division operator will return those tuples from relation A which are associated to every B’s tuple.

DIVISION

A :

SNo	PNo
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

B1 :

PNo
P2

B2 :

PNo
P2
P4

B3 :

PNo
P1
P2
P3

A/B1 :

SNo
S1
S2
S3
S4

A/B2 :

A/B3 :

Consider two instances A & B in which A has two fields X &Y and B has just one field Y with same domain as A.

Then, we define division operation **A/B** as set of all X values such that for every Y in B there is a tuple (X,Y) in A.

NOTE: The attribute which divides i.e. B1 will not appear in the result after division.
Find matching with B1.

DIVISION

A :

SNo	PNo
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4

B1 :

PNo
P2

B2 :

PNo
P2
P4

B3 :

PNo
P1
P2
P3

A/B1 :

SNo
S1
S2
S3
S4

A/B2 :

SNo
S1
S4

A/B3 :

SNo
S1

Consider two instances A & B in which A has two fields X &Y and B has just one field Y with same domain as A.

Then, we define division operation **A/B** as set of all X values such that for every Y in B there is a tuple (X,Y) in A.

NOTE: The attribute which divides i.e. B1 will not appear in the result after division.
Find matching with B1.

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 ⋈.

TYPES OF JOINS

- CONDITIONAL JOIN OR THETA JOIN
- EQUI JOIN
- NATURAL JOIN
- OUTER JOIN
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

CONDITIONAL JOIN ($\bowtie_c$) or THETA JOIN ($\bowtie_\theta$)

Conditional Join is used when you want to join two or more relation based on some conditions.

STUDENT -

Roll_no	Sname	Address	S_age
101	D	7	15
102	L	8	16
103	S	10	17

EMPLOYEE -

Emp_no	Ename	E_age
101	101	25
102	103	26

Example: Select students whose Roll_no is greater than Emp_no of employees.

STUDENT $\bowtie_c$ $_{\text{STUDENT.Roll\_no>EMPLOYEE.Emp\_no}}$ EMPLOYEE

EQUI JOIN

Equi join is a **special case of conditional join** *where only **equality** condition holds* between a pair of attributes.

STUDENT -

Roll_no	Sname	Address	S_age
101	D	7	15
102	L	8	16
103	S	10	17

EMPLOYEE -

Emp_no	Ename	E_age
101	101	25
102	103	26

Example: Select students whose Roll_no is equal to Emp_no of employees.

$$\textbf{STUDENT} \bowtie_{\textbf{STUDENT.ROLL\_NO=EMPLOYEE.EMP\_NO}} \textbf{EMPLOYEE}$$

NATURAL JOIN (⋈)

- It is a special case of equijoin in which ***equality condition hold on all attributes which have same name in relations R and S*** (relations on which join operation is applied).
- While applying natural join on two relations, there is no need to write equality condition explicitly.
- Natural Join will also return the similar attributes only once as their value will be same in resulting relation.
- If the two relations have no attributes in common then Natural Join will simply work as a cross product.

NATURAL JOIN (⋈)

Sailor -

Sid	Sname	Rating	Age
22	D	7	44
31	L	8	55
58	S	10	35

Boat -

Sid	Bid	Day
22	101	10/10/96
58	103	11/12/96

Example: *Select sailors whose Sid is equal to Sid of Boat.*

SAILOR ⋈ BOAT

NOTE: While applying natural join on two relations, there is no need to write equality condition explicitly.

LEFT OUTER JOIN(⟕)

- When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Left Outer Joins **gives all tuples of R** in the result set. The tuples of R which do not satisfy join condition will have values as NULL for attributes of S.

R :

A	B	C
1	2	3
4	5	6
7	8	9

S :

C	D
3	7
10	11

R ⟕ S:

A	B	C	D
1	2	3	7
4	5	6	NULL
7	8	9	NULL

RIGHT OUTER JOIN(⋊)

- When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Right Outer Joins **gives all tuples of S** in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R.

R :

A	B	C
1	2	3
4	5	6
7	8	9

S :

C	D
3	7
10	11

R ⋊ S:

A	B	C	D
1	2	3	7
NULL	NULL	10	11

FULL OUTER JOIN(⟗)

- When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions.
- But Full Outer Joins **gives all tuples of S and all tuples of R** in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R and vice versa.

R :

A	B	C
1	2	3
4	5	6
7	8	9

S :

C	D
3	7
10	11

R ⟗ S:

A	B	C	D
1	2	3	7
4	5	6	NULL
7	8	9	NULL
NULL	NULL	10	11

Exercise

Ques. *Select students whose ROLL_NO is greater than EMP_NO of employees and details of other students as well*

Ques. *Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well*

Ques. *Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well and other Students as well*

Exercise

- **LEFT OUTER JOIN (⟕)**

Example: *Select students whose ROLL_NO is greater than EMP_NO of employees and details of other students as well*

$$\textbf{STUDENT} ⟕_{\textbf{STUDENT.ROLL\_NO>EMPLOYEE.EMP\_NO}} \textbf{EMPLOYEE}$$

- **RIGHT OUTER JOIN (⟖)**

Example: *Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well*

$$\textbf{STUDENT} ⟖_{\textbf{STUDENT.ROLL\_NO>EMPLOYEE.EMP\_NO}} \textbf{EMPLOYEE}$$

- **FULL OUTER JOIN (⟗)**

Example: *Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well and other Students as well*

$$\textbf{STUDENT} ⟗_{\textbf{STUDENT.ROLL\_NO>EMPLOYEE.EMP\_NO}} \textbf{EMPLOYEE}$$

RELATIONAL CALCULUS

- 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

- ➢ TUPLE RELATIONAL CALCULUS
- ➢ DOMAIN RELATIONAL CALCULUS

TUPLE RELATIONAL CALCULUS

- 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:

{T | P (T)}   or {T | Condition (T)}

where,

T is the resulting tuples

P(T) is the condition used to fetch T.

TUPLE RELATIONAL CALCULUS

{T | P (T)}   or {T | Condition (T)}

where,

T is the resulting tuples

P(T) is the condition used to fetch T.

For example:

{ T.name | Author(T) AND T.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'.

TUPLE RELATIONAL CALCULUS

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

For example:

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

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

DOMAIN RELATIONAL CALCULUS

- 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:

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

DOMAIN RELATIONAL CALCULUS

Notation:

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

where

a1, a2 are attributes
P stands for formula built by inner attributes

For example:

{< article, page, subject > | ∈ cse ∧ subject = 'database'}

Output: This query will yield the article, page, and subject from the relational cse, where the subject is a database.

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 the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

Types of Normalization

- 1 NF
- 2 NF
- 3 NF
- BCNF