RELATIONAL ALGEBRA

Course Code: **INT306**

Contents

- What is Relational Algebra?
- Operations in Relational Algebra
- Extended Relational Algebra Operations

Relational Algebra

- The **relational algebra** is a **procedural query language**
- It consists of a set of operations that take one or two relations as input and produce a new relation as their result.
- These operations enable a user to specify basic **retrieval requests (or queries)**

Cont.

- The fundamental operations in the relational algebra are *select, project, union, set difference, Cartesian product, and rename*

- The **select, project, and rename** operations are called *unary operations, because* they operate on one relation

- The other three operations operate on pairs of relations and are, therefore, called *binary operations*

What is a Query Language?

- Database language.
- Use for retrieving informations from database.
- Two types:-

  - Procedural

    Specifies what data are needed & how to get those data.

  - Non-procedural

    Specifies what data are needed except how to get those data.

Basic concepts

Domain: There are set of permitted values for every attribute, called its domain.

Exp- Domain of roll number{10,11,23,56,78}

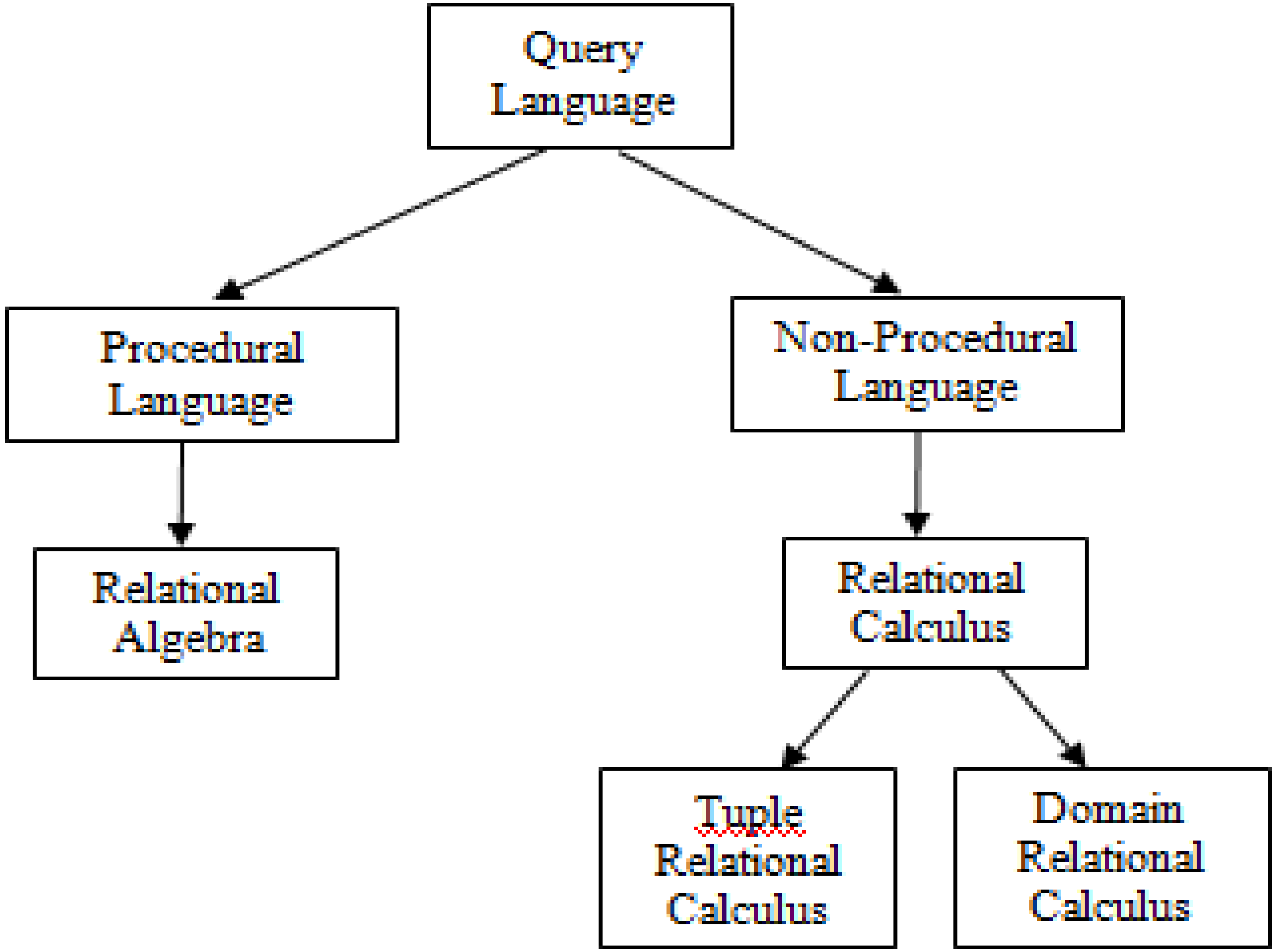
Domain of branch{CSE,IT,ME,ECE}

Tuple: Each row in a relation is called tuple.

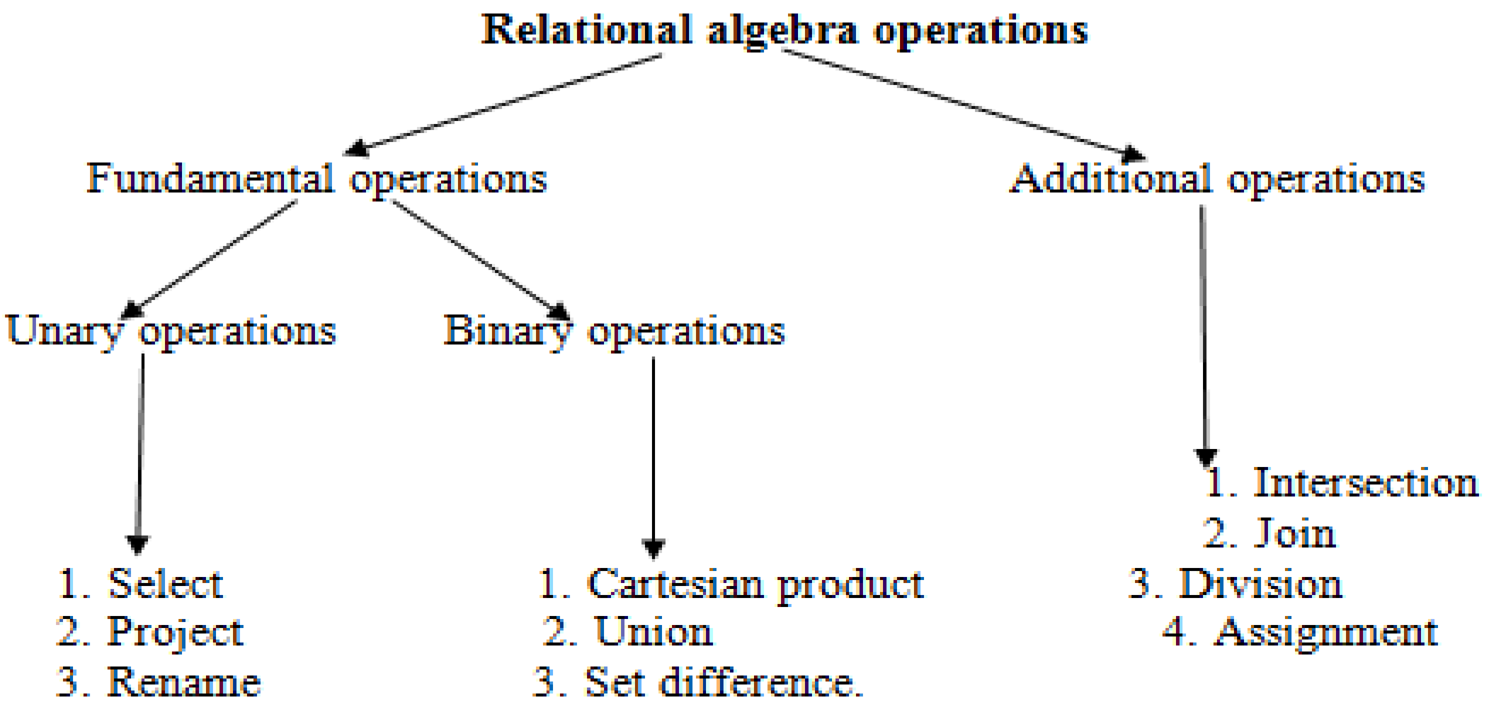
Relation: Collection of homogeneous tuples.

Degree or Arity: Number of attributes in relation R.

Cardinality: Number of tuples in relation R.


Query Language
Procedural Language
Non-Procedural Language
Relational Algebra
Relational Calculus
Tuple Relational Calculus
Domain Relational Calculus

INTRODUCTION



Relational algebra operations

Fundamental operations

Additional operations

Unary operations

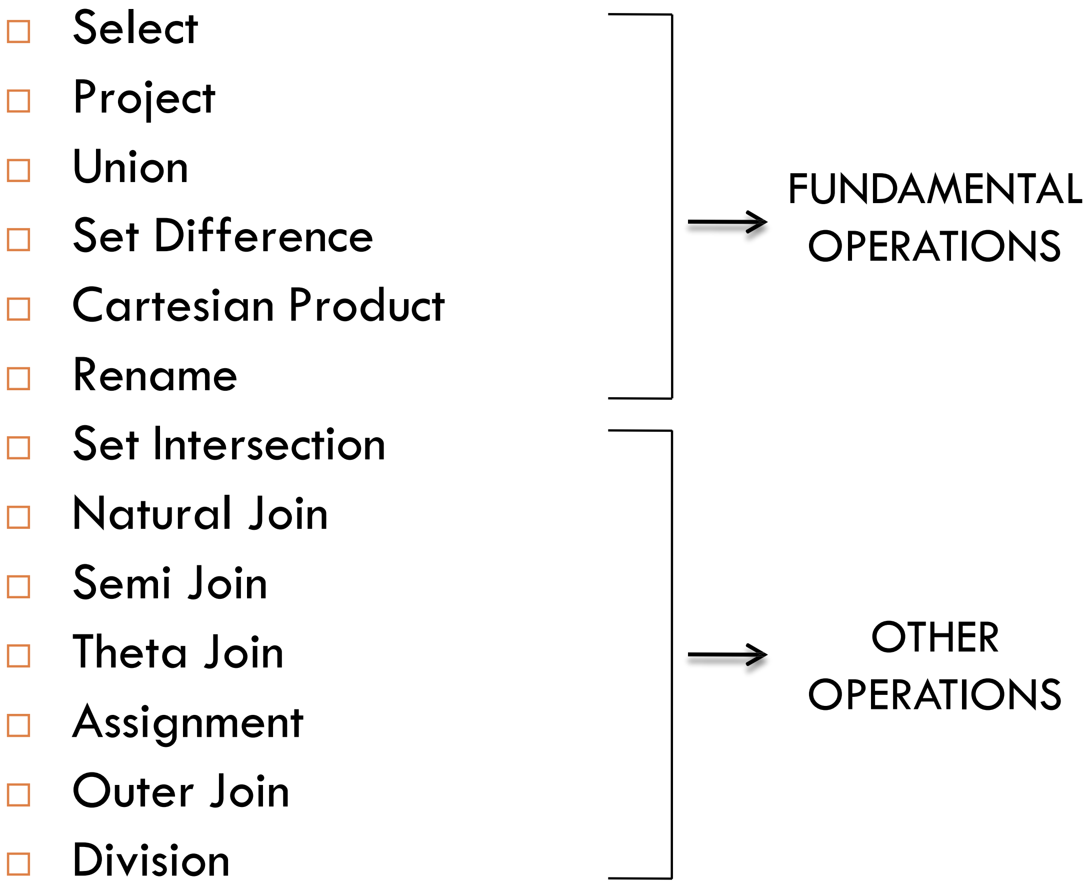
Binary operations

1. Select
2. Project
3. Rename

1. Cartesian product
2. Union
3. Set difference.

1. Intersection
2. Join
3. Division
4. Assignment

Operations



- □ Select
- □ Project
- □ Union
- □ Set Difference
- □ Cartesian Product
- □ Rename

→ FUNDAMENTAL OPERATIONS

- □ Set Intersection
- □ Natural Join
- □ Semi Join
- □ Theta Join
- □ Assignment
- □ Outer Join
- □ Division

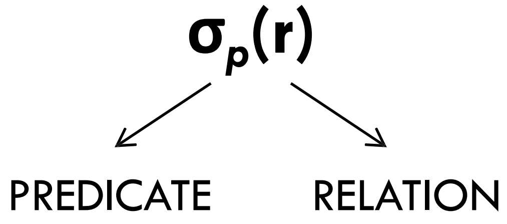
→ OTHER OPERATIONS

Unary Relational Operations

- SELECT (symbol: σ (sigma))
  - Selects a subset of rows from relation
- PROJECT (symbol: π (pi))
  - Selects columns from relation
- RENAME (symbol: ρ (rho))

Select Operation (σ)

- Unary operation.
- Selects tuples *(rows)* that satisfy the given predicate *(condition)* from a relation *(table)*.



$$\sigma_{p}(\mathbf{r})$$

PREDICATE RELATION

- There can be more than one predicate connected by connectors (*and* [**^**], *or* [**v**], *not* [¬]).
- Comparisons are performed using relational operators (**=, ≠, ≥, <, >, ≤**).

Examples

person(*name*, *age*, *weight*)

1. Find the details of all persons having age greater than equals to 34.

Ans: $\sigma_{age \geq 34}$(person)

2. Find the details of all persons having age greater than 34 & weight equals to 54.

Ans: $\sigma_{age > 34 \wedge weight = 54}$(person)

Illustrations

person

name	age	weight
Harry	34	80
Sally	28	64
George	29	70
Helena	54	54
Peter	34	80

$б_{age>=34}(person)$

name	age	weight
Harry	34	80
Helena	54	54
Peter	34	80

$б_{age>34\ \wedge\ weight=54}(person)$

name	age	weight
Helena	54	54

Project Operation (Π)

- Unary operation.
- Selects specified attributes (*columns*) from a relation.

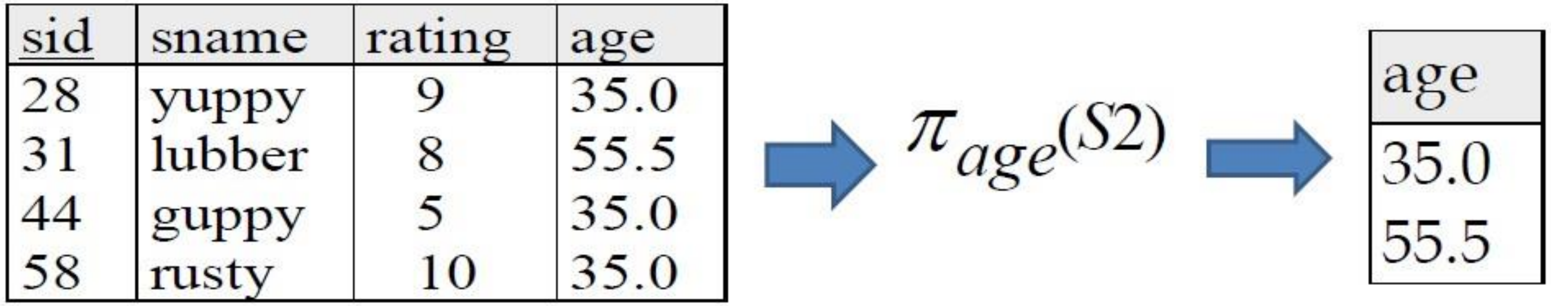
$$\boldsymbol{\Pi}_{\underbrace{\mathbf{A1,\ A2,\ An}}_{\text{ATTRIBUTES}}}(\underset{\text{RELATION}}{\mathbf{r}})$$

PROJECTION

$$\pi_{fields}(Input)$$

Allows us to extract columns from a relation

Example:



sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$\rightarrow \pi_{age}(S2) \rightarrow$

age
35.0
55.5

Example

employee(*name, age, salary*)

1. Find the name of all employees.

Ans: Π_{name}(employee)

employee

name	age	salary
Harry	34	80,000
Sally	28	90,000
George	29	70,000
Helena	54	54,280
Peter	34	40,000

$\Pi_{\textbf{name}}$(employee)

name
Harry
Sally
George
Helena
Peter

EXAMPLE

- Example: To list each employee's first and last name and salary, the following is used:

$$\pi_{\text{LNAME, FNAME,SALARY}}(\text{EMPLOYEE})$$

Composition of Relational Operations

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

sname	rating
yuppy	9
rusty	10

Example

- “Find the name of all instructors in the Physics department.

π *name* (σ *dept name =“Physics”* *(instructor))*

Combination of Select & Project Operations - Example

employee(*name*, *age*, *salary*)

1. Find the name of employees earning more than 80,000

Ans: $\Pi_{name}(\sigma_{salary>80000}(employee))$

employee

name	age	salary
Harry	34	80,000
Sally	28	90,000
George	29	70,000
Helena	54	54,280
Peter	34	40,000

б$_{salary>80,000}$(employee)

name	age	salary
Sally	28	90,000

∏ $_{name}$(б$_{salary>80,000}$(employee))

name
Sally

Rename Operation (ρ)

- □ Unary operation

$$\rho_{a / b}(r)$$

where,

- **r** is a relation

- **a** and **b** are attributes

- □ The result is identical to **r** except that the **b** field in the relation is renamed to an **a** field.

Example

employee

name	**age**	**salary**
Harry	34	80,000
Sally	28	90,000
George	29	70,000
Helena	54	54,280
Peter	34	40,000

$\rho_{\text{ename / name}}(\text{employee})$

ename	**age**	**salary**
Harry	34	80,000
Sally	28	90,000
George	29	70,000
Helena	54	54,280
Peter	34	40,000

Relational Algebra Operations
Set Theory

- Union
- Intersection
- Set Difference / Minus
- Cartesian Product

Union Operation (U)

- □ Binary operation.
- □ Returns the union of two compatible relations (say **r** & **s**).

r U s

where,

- **r** & **s** must have the same number of attributes.

- Attribute domains must be compatible.

- Duplicate tuples are automatically eliminated.

UNION

- The two operand relations R and S must be "type compatible" (or UNION compatible)

- Two relations are *union compatible* if
  - Relation R and S should have same arity, Both have same number of columns
  - Names of attributes and the domain type are the same in both

Example

course(*name*, *semester*, *teacher*)

1. Find the name of all courses taught in the 1st semester, the 2nd semester, or both.

Ans: $\Pi_{\text{name}}(\sigma_{\text{semester=“1st"}}(\text{course}))$

$\cup$

$\Pi_{\text{name}}(\sigma_{\text{semester=“2nd"}}(\text{course}))$

Illustration

course

name	semester	teacher
C1	1st	T1
C2	2nd	T2
C3	3rd	T3
C4	4th	T4
C1	2nd	T1

$\Pi_{name}(\text{б}_{semester=\text{“1st"}}(course))$

name
C1

$\Pi_{name}(\text{б}_{semester=\text{“2nd"}}(course))$

name
C2
C1

$\Pi_{name}(\text{б}_{semester=\text{“1st"}}(course)) \cup \Pi_{name}(\text{б}_{semester=\text{“2nd"}}(course))$

name
C1
C2

Example

Tables:

Person (*SSN, Name, Address, Hobby*)

Professor (*Id, Name, Office, Phone*)

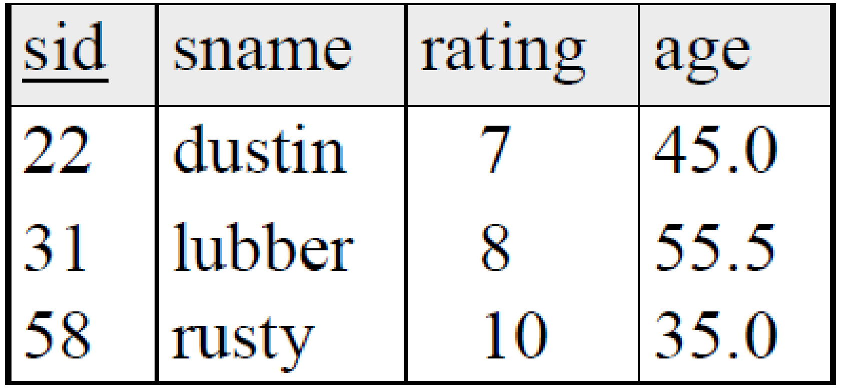
are not union compatible. However

$$\Pi_{Name}\,(\text{Person})\text{ and }\Pi_{Name}\,(\text{Professor})$$

are union compatible and

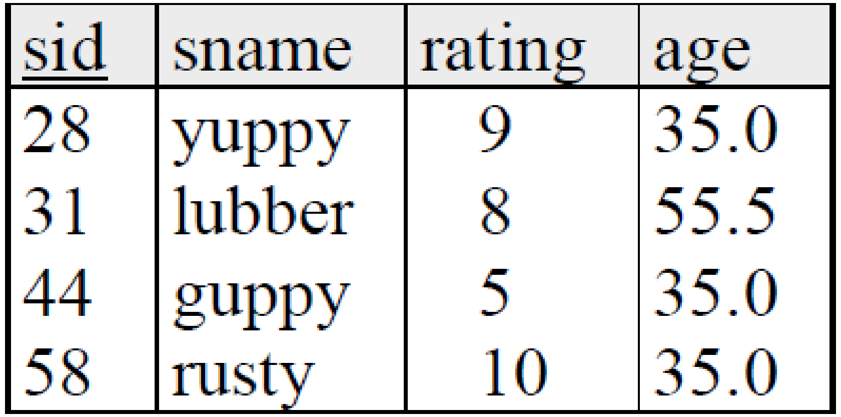
Set Operation: Union

S1

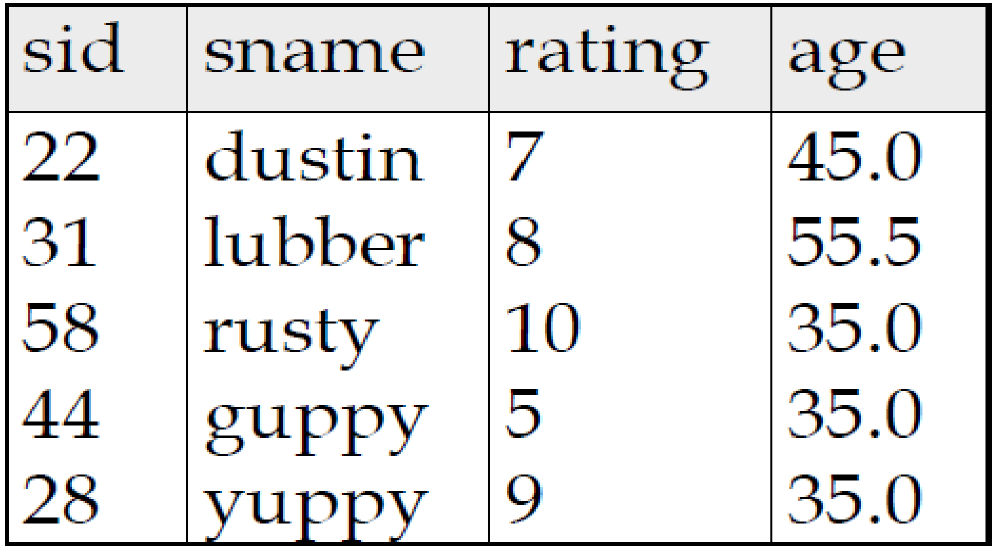


sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2



sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0



sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

S1 U S2

Set Difference (-)

- Binary operation.
- Returns the tuples which are present in first relation (**r**) but are not in the second relation (**s**).

r – s

- Like **union** the conditions are same for a valid **r – s** operation.

Example

course(*name*, *semester*, *teacher*)

1. Find the name of all courses taught in the 1st semester but not in 2nd semester.

Ans: $\Pi_{name}(\sigma_{semester=\text{“1st"}}(course))$

$-$

$\Pi_{name}(\sigma_{semester=\text{“2nd"}}(course))$

Illustration

course

name	semester	teacher
C1	1st	T1
C2	2nd	T2
C3	1st	T3
C4	4th	T4
C1	2nd	T1

$\Pi_{name}(\text{б}_{semester=\text{“1st"}}(course))$

name
C1
C3

$\Pi_{name}(\text{б}_{semester=\text{“2nd"}}(course))$

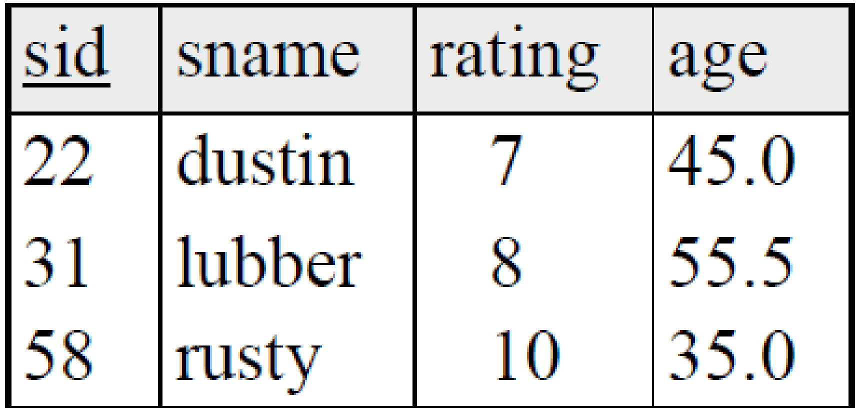
name
C2
C1

$\Pi_{name}(\text{б}_{semester=\text{“1st"}}(course)) - \Pi_{name}(\text{б}_{semester=\text{“2nd"}}(course))$

name
C3

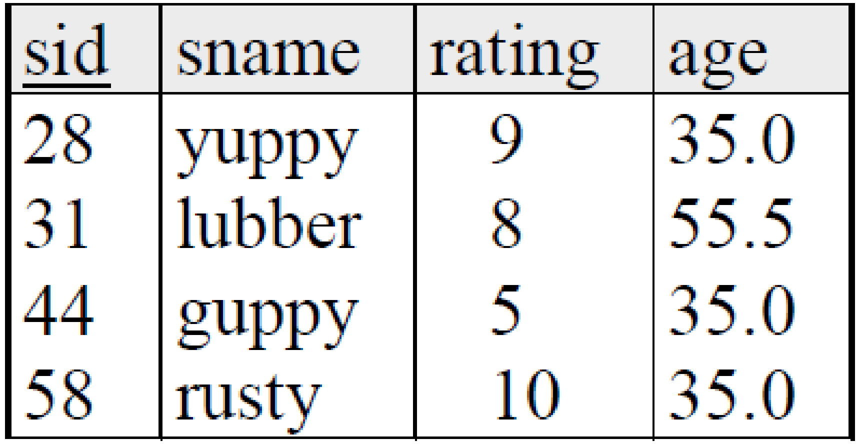
Set Operation: Set-Difference

S1



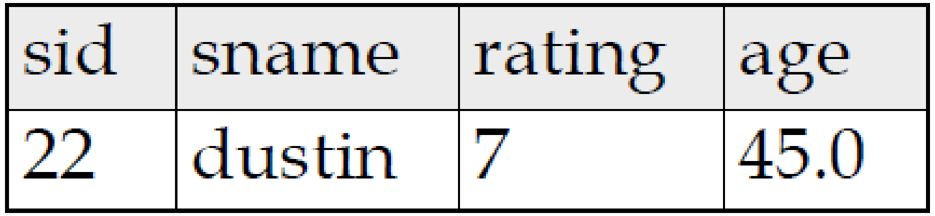
sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2



sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S1 – S2



sid	sname	rating	age
22	dustin	7	45.0

Set Intersection (∩)

- Binary operation.

$$\mathbf{r} \cap \mathbf{s}$$

- Defines a relation consisting of the set of all tuples that are in both **r** and **s.**
- Like **union** the conditions are same for a valid **r ∩ s** operation.
- Expressed using basic operations:

$$\mathbf{r} \cap \mathbf{s} = \mathbf{r} - (\mathbf{r} - \mathbf{s})$$

Example

course(*name*, *semester*, *teacher*)

1. Find the name of all courses taught in both the 1st & 2nd semesters.

Ans: $\Pi_{name}(\sigma_{semester=\text{“1st"}}(course))$

$\cap$

$\Pi_{name}(\sigma_{semester=\text{“2nd"}}(course))$

Illustration

course

name	semester	teacher
C1	1st	T1
C2	2nd	T2
C3	1st	T3
C4	4th	T4
C1	2nd	T1

$\prod_{\text{name}}(\text{б}_{\text{semester=“1st"}}(\text{course}))$

name
C1
C3

$\prod_{\text{name}}(\text{б}_{\text{semester=“2nd"}}(\text{course}))$

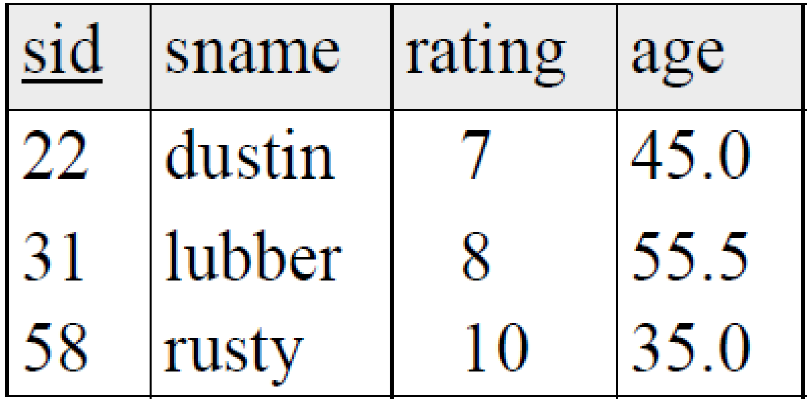
name
C2
C1

$\prod_{\text{name}}(\text{б}_{\text{semester=“1st"}}(\text{course})) \cap \prod_{\text{name}}(\text{б}_{\text{semester=“2nd"}}(\text{course}))$

name
C1

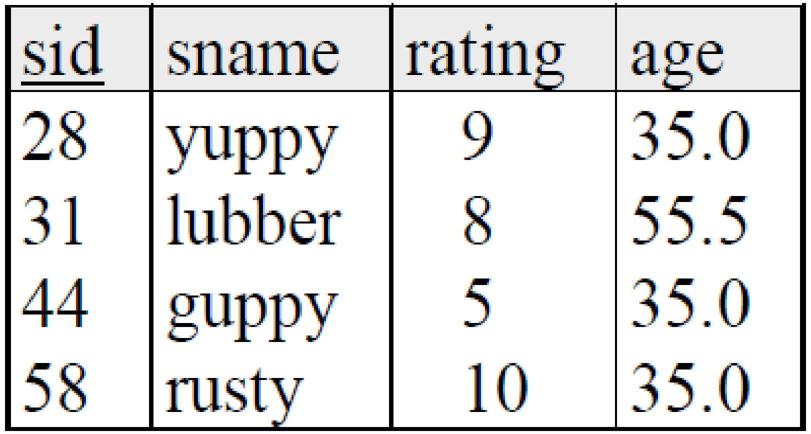
Set Operation: Intersection

S1

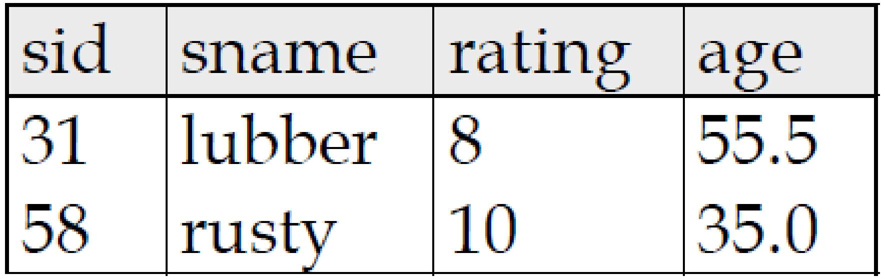


sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2



sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0



sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

S1 ∩ S2

Cartesian Product (x)

- Binary operation.
- Combines information from any two relations (**r** & **s**).

  r x s
- It defines a relation by concatenating every tuple of relation **r** with every tuple of relation **s.**

Illustration

person

name	age	weight
Harry	34	80
Sally	28	64
George	29	70

city

city
San Jose
Austin

person X city

name	age	weight	city
Harry	34	80	San Jose
Harry	34	80	Austin
Sally	28	64	San Jose
Sally	28	64	Austin
George	29	70	San Jose
George	29	70	Austin

SQL Built in Functions

NUMERIC FUNCTIONS:

- abs(n) – returns absolute value of n

  SQL> select abs(-2) from dual; - 2
- ceil(n) – ceil(3.78) - 4
- floor(n) – floor(2.4) - 2
- sqrt(n)
- power(n,m)

  SQL> select power(5,2) from dual; - 25
- mod(a,b)
- cos(n)

SQL Built in Functions

String/Character FUNCTIONS:

- initcap(string)

  SQL > select initcap(‘hello’) from dual; - Hello

- upper(string)
- lower(string)
- ltrim(string)

  SQL > select ltrim(‘   hello’) from dual; - hello

- rtrim(string)
- replace( )

  SQL > select replace(‘hello’, ’ll’, ’r’) from dual; - hero

- length( )

  SQL > select length(‘hello’) from dual; - 5