**TRIBHUVAN UNIVERSITY**
**SAMRIDDHI COLLEGE**
**Lokanthali-16, Bhaktapur, Nepal**

**Bachelor of Science in**
**Computer Science & Information Technology**
**(B.Sc. CSIT)**
**Fourth Semester**

# Unit 5. *The Relational Algebra and the Relational Calculus*

## Session 1

by:
**Er. Bikal Adhikari**
(*B.E., M.Sc. Engg., M.Sc. ISE*)

Date:- 27th Mar., 2023

# Unit 5. The Relational Algebra and the Relational Calculus

**5 hours**

Unary Relational Operations: SELECT and PROJECT; Relational Algebra Operations from Set Theory; Binary Relational Operations: JOIN and DIVISION; Additional Relational Operations; the Tuple Relational Calculus; the Domain Relational Calculus

# WHAT IS RELATIONAL ALGEBRA ?

→ Relational algebra defines the theoretical way of manipulating table contents using the relational functions.

→ It is a procedural query language.

→ Consists of set of operations.

→ Takes one(unary) or two(binary) relations as input and produce a new relation as output.

→ Introduced by E.F. Codd in 1970 as a basis for a database query languages.

# What is a Query Language?

- → Database language
- → Used for retrieving informations from database.
- → Two types:
  - ▪ **Procedural**
    - • Specifies what data are needed and how to get those data.
  - ▪ **Non-Procedural:**
    - • Specifies what data are needed except how to get those data.

# OPERATIONS

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

***Fundamental Operations***

- Natural Join
- Theta Join
- Equi Join
- Assignment
- Outer Join
- Division

***Other Operations***

# SELECT OPERATIONS(σ)

- Unary Operation
- Select ***tuples*** (rows) that satisfy the given ***predicate***(condition) from a ***relation***(table)

$$\sigma_P(r)$$

Predicate　　　Relation

- There can be more than one predicate connected by connectors (and [Ʌ], or [V], not [¬]).
- Comparisons are performed by using relational operators

  (=, ≠ , ≥, ≤, >, <, etc)

# EXAMPLES

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

Example 1: Find the details of all persons having age greater than or equal to 34.

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

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

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

# Illustrations

***person***

| ***name*** | ***age*** | ***weight*** |
|---|---|---|
| Ram | 34 | 80 |
| Shyam | 28 | 64 |
| Hari | 29 | 70 |
| Rita | 54 | 54 |
| Sita | 34 | 80 |

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

| ***name*** | ***age*** | ***weight*** |
|---|---|---|
| Ram | 34 | 80 |
| Rita | 54 | 54 |
| Sita | 34 | 80 |

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

| ***name*** | ***age*** | ***weight*** |
|---|---|---|
| Rita | 54 | 54 |

# PROJECT OPERATION(π)

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

$$\Pi_{\underbrace{A1, A2, \ldots, An}_{\text{Attributes}}}(r)$$

Relation

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

Example 1: Find the names of all employees

**Answer:** $\prod_{name}(employee)$

***employee***

| ***name*** | ***age*** | ***salary*** |
|---|---|---|
| Ram | 34 | 80,000 |
| Shyam | 28 | 90,000 |
| Hari | 29 | 70,000 |
| Rita | 54 | 54,280 |
| Sita | 34 | 40,000 |

$\prod_{name}(employee)$

| ***name*** |
|---|
| Ram |
| Shyam |
| Hari |
| Rita |
| Sita |

# COMBINATION OF SELECT AND PROJECT OPERATION

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

Example 1: Find the names of all employees earning more than 80,000.

**Answer:** $\prod_{name}\left(\sigma_{salary>80000}(employee)\right)$

***employee***

| ***name*** | ***age*** | ***salary*** |
|---|---|---|
| Ram | 34 | 80,000 |
| Shyam | 28 | 90,000 |
| Hari | 29 | 70,000 |
| Rita | 54 | 54,280 |
| Sita | 34 | 40,000 |

$\sigma_{salary>80,000}(employee)$

| ***name*** | ***age*** | ***salary*** |
|---|---|---|
| Shyam | 28 | 90,000 |

$\prod_{name}\left(\sigma_{salary>80000}(employee)\right)$

| ***name*** |
|---|
| Shyam |

# UNION OPERATION(∪)

→ Binary Operation.

→ Returns the union of two compatible relations (say r & s).

**r ∪ s**

Where,

- r & s must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

# EXAMPLE:

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

Example: Find the name of all courses taught in the 1st semester, the second semester, or both.

Answer:

$$\prod_{name}\left(\sigma_{semester="1st"}(course)\right)$$

$$\cup$$

$$\prod_{name}\left(\sigma_{semester="2nd"}(course)\right)$$

# ILLUSTRATION

***course***

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

$$\prod_{name}(\sigma_{semester="1st"}(course))$$

| *name* |
|---|
| C1 |

$$\prod_{name}(\sigma_{semester="2nd"}(course))$$

| *name* |
|---|
| C2 |
| C1 |

$$\prod_{name}(\sigma_{semester="1st"}(course)) \cup \prod_{name}(\sigma_{semester="2nd"}(course))$$

| *name* |
|---|
| C1 |
| C2 |

# Set Intersection(∩)

- Binary Operation.

$$r \cap s$$

- 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 $r \cap s$ operation.
- Expressed using basic operations: $r \cap s = r - (r\text{-}s)$

# EXAMPLE:

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

Example: Find the name of all courses taught in the 1st semester and second semester.

Answer:

$$\prod_{name}\left(\sigma_{semester="1st"}(course)\right)$$

$$\cap$$

$$\prod_{name}\left(\sigma_{semester="2nd"}(course)\right)$$

# ILLUSTRATION

***course***

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

$$\prod_{name}\left(\sigma_{semester="1st"}(course)\right)$$

| ***name*** |
|---|
| C1 |
| C3 |

$$\prod_{name}\left(\sigma_{semester="2nd"}(course)\right)$$

| ***name*** |
|---|
| C2 |
| C1 |

$$\prod_{name}\left(\sigma_{semester="1st"}(course)\right)\cap\prod_{name}\left(\sigma_{semester="2nd"}(course)\right)$$

| ***name*** |
|---|
| C1 |

# SET DIFFERENCE(-)

→ Binary Operation.

→ Returns the tuples which are present in first relation (r) but are not present in the second relation (s).

r-s

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

# EXAMPLE:

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

Example: Find the name of all courses taught in the 1st semester, but not in the second semester

***Answer:***

$$\prod_{name}\left(\sigma_{semester="1st"}(course)\right)$$

$$-$$

$$\prod_{name}\left(\sigma_{semester="2nd"}(course)\right)$$

# ILLUSTRATION

***course***

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

$$\prod_{name}(\sigma_{semester="1st"}(course))$$

| *name* |
|---|
| C1 |
| C3 |

$$\prod_{name}(\sigma_{semester="2nd"}(course))$$

| *name* |
|---|
| C2 |
| C1 |

$$\prod_{name}(\sigma_{semester="1st"}(course))-\prod_{name}(\sigma_{semester="2nd"}(course))$$

| *name* |
|---|
| C3 |

# CARTESIAN PRODUCT(×)

- → Binary Operation.
- → Returns the union of two compatible relations (say r & s).

$$\mathbf{r \times s}$$

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

# ILLUSTRATION

***person***

| ***name*** | ***age*** | ***weight*** |
|---|---|---|
| Ram | 34 | 80 |
| Shyam | 28 | 64 |
| Hari | 29 | 70 |

***city***

| |
|---|
| Pokhara |
| Kathmandu |

***person × city***

| ***name*** | ***age*** | ***weight*** | ***city*** |
|---|---|---|---|
| Ram | 34 | 80 | Pokhara |
| Ram | 34 | 80 | Kathmandu |
| Shyam | 28 | 64 | Pokhara |
| Shyam | 28 | 64 | Kathmandu |
| Hari | 29 | 70 | Pokhara |
| Hari | 29 | 70 | Kathmandu |

# 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*** |
|---|---|---|
| Ram | 34 | 80,000 |
| Shyam | 28 | 90,000 |
| Hari | 29 | 70,000 |
| Rita | 54 | 54,280 |
| Sita | 34 | 40,000 |

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

| ***ename*** | ***age*** | ***salary*** |
|---|---|---|
| Ram | 34 | 80,000 |
| Shyam | 28 | 90,000 |
| Hari | 29 | 70,000 |
| Rita | 54 | 54,280 |
| Sita | 34 | 40,000 |

# JOIN

→ **Join** is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

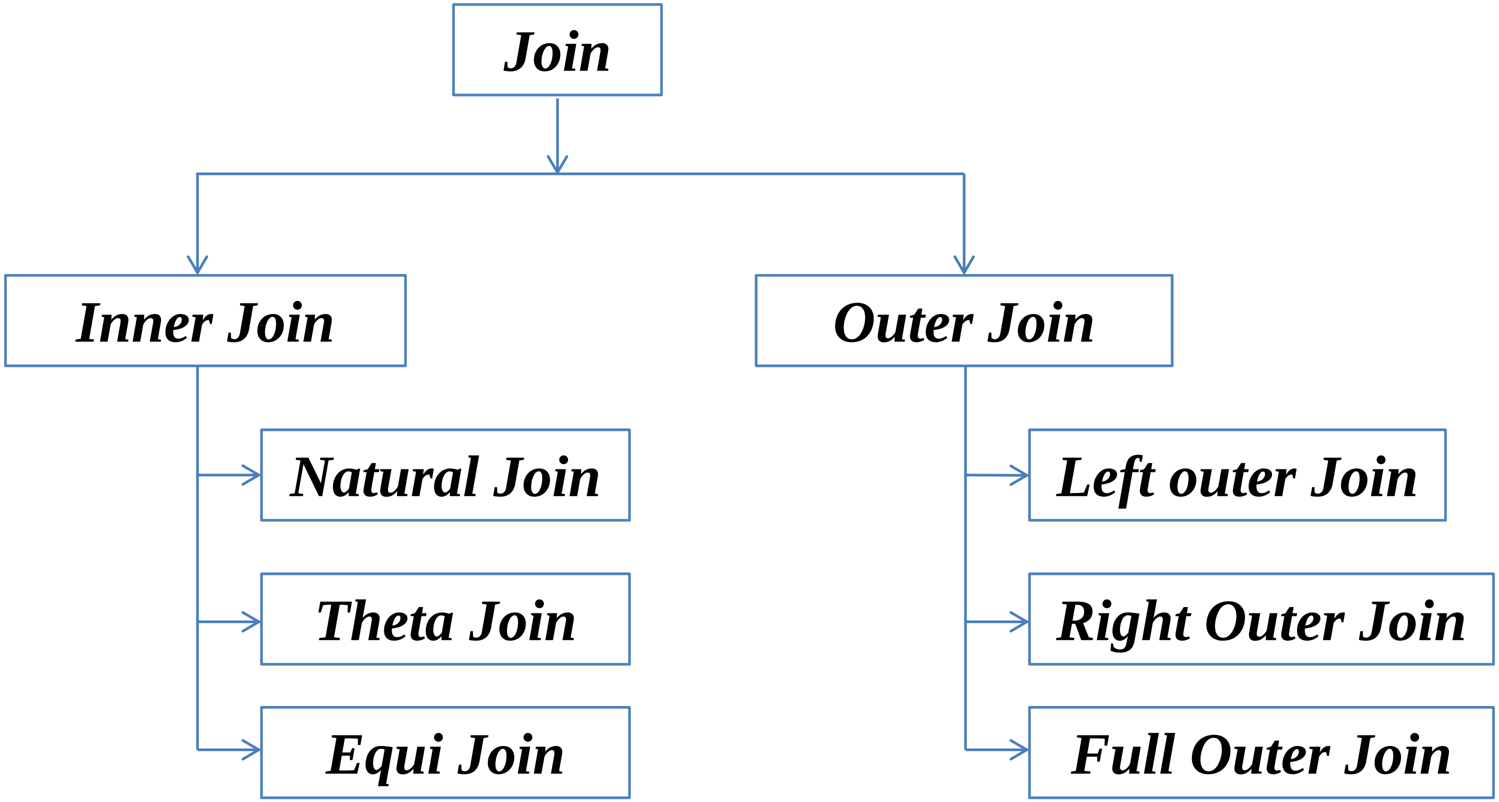


Figure: ***Types of join***

# Natural Join(⋈)

→ Natural join does not use any comparison operator.

→ It does not concatenate the way a Cartesian product does.

→ We can perform a Natural Join only if there is at least one common attribute that exists between two relations.

→ In addition, the attributes must have the same name and domain.

→ Natural join acts on those matching attributes where the values of attributes in both the relations are same.

*s*

| ***a*** | ***b*** |
|---|---|
| a1 | b1 |
| a2 | b2 |
| a3 | b3 |

*r*

| ***b*** | ***c*** |
|---|---|
| b1 | c1 |
| b2 | c2 |
| b3 | c3 |

$s \bowtie r$

| ***b*** | ***a*** | ***c*** |
|---|---|---|
| b1 | a1 | c1 |
| b2 | a2 | c2 |
| b3 | a3 | c3 |

$r \bowtie s$

| ***b*** | ***c*** | ***a*** |
|---|---|---|
| b1 | c1 | a1 |
| b2 | c2 | a2 |
| b3 | c3 | a3 |

# THETA JOIN OR CONDITION JOIN($\bowtie_{\Theta}$)

→ Variant of the natural join.

→ Combine a selection and a cartesian product into a single operation.

$$\mathbf{r \bowtie_{\Theta} s = \sigma_{\Theta}(r \times s)}$$

Where,

r & s are relations.

$\Theta$ is a predicate.

# ILLUSTRATION

*Students*

| stud# | name | course |
|---|---|---|
| 100 | Fred | PH |
| 200 | Dave | CM |
| 300 | Bob | CM |

*Courses*

| course# | name |
|---|---|
| PH | Pharmacy |
| CM | Computing |

*Students* $\bowtie_{\text{stud\#=200}}$ **Courses**

| stud# | name | course | Course# | Courses.name |
|---|---|---|---|---|
| 200 | Dave | CM | PH | Pharmacy |
| 200 | Dave | CM | CM | Computing |

# Equi Join

- → When Theta join uses only **equality** comparison operator, it is said to be equijoin.
- → Illustration on equi join is given in the next slide.

# Illustration on Equi Join

*Students*

| stud# | name | course |
|---|---|---|
| 100 | Fred | PH |
| 200 | Dave | CM |
| 300 | Bob | CM |

*Courses*

| course# | name |
|---|---|
| PH | Pharmacy |
| CM | Computing |

*Students* $\bowtie_{\textbf{course =course\#}}$ **Courses**

| stud# | name | course | course# | Courses.name |
|---|---|---|---|---|
| 100 | Fred | PH | PH | Pharmacy |
| 200 | Dave | CM | CM | Computing |
| 300 | Bob | CM | CM | Computing |

***Thank You !!!***