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 2

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

Assignment Operation

- → Writing a relational algebra expression by assigning parts of it to temporary relation variables.
- → Works like assignment in a programming language.
- → Denoted by ←

Illustration of Assignment in Intersection

course

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

$$r1 \leftarrow \prod_{name}(\sigma_{semester="1st"}(course))$$

name
C1
C3

$$r2 \leftarrow \prod_{name}(\sigma_{semester="2nd"}(course))$$

name
C2
C1

$$r1 \cap r2$$

name
C1

Outer Join

→ Deals with missing information.

→ Works in a similar fashion like natural join but preserves those tuples that would be lost in the join by creating tuples in the result containing null values.

→ Three types:

- Left outer join
- Right outer join
- Full outer join

LEFT OUTER JOIN

- → Takes all the tuples in the left relation that did not match with any tuple in the right relation.
- → Pads the tuples with null values for all other attributes from the right relation.
- → Adds them to the result of the natural join.

$$r ⟕ s$$

- → Where,

  r and s are relations.

ILLUSTRATIONS

r

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

s

b	***c***
b1	c1
b2	c2
b4	c4

r* ⋈ *s

a	***b***	***c***
a1	b1	c1
a2	b2	c2
a3	b3	*null*

RIGHT OUTER JOIN

- → Takes all the tuples in the right relation that did not match with any tuple in the left relation.
- → Pads the tuples with null values for all other attributes from the left relation.
- → Adds them to the result of the natural join.

$$r ⟖ s$$

- → Where,

  r and s are relations.

ILLUSTRATIONS

r

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

s

b	***c***
b1	c1
b2	c2
b4	c4

r* ⟕ *s

a	***b***	***c***
a1	b1	c1
a2	b2	c2
null	b4	*c4*

FULL OUTER JOIN

→ Performs both left and right outer join operations.

→ Pads the tuples with null values from the right relation that did not match with any from the left relation and vice versa.

Left Outer Join + Right Outer Join = Full Outer Join

$$r \bowtie s$$

→ Where,

r and s are relations.

ILLUSTRATIONS

r

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

s

b	***c***
b1	c1
b2	c2
b4	c4

r* ⋈ *s

a	***b***	***c***
a1	b1	c1
a2	b2	c2
a3	b3	null
null	b4	*c4*

DIVISION OPERATION

→ Binary operation

r / s

Where

- r is the dividend relation and s is the divisor relation.
- All the attributes in s also appear in r.
- s is not empty.

→ The result consists of the restrictions of tuples in r to the attribute names unique to r, i.e. in the header of r but not in the header of s, for which it holds that all their combination with tuples in s are present in r.

ILLUSTRATION

completed

student	*task*
Fred	Database1
Fred	Database2
Fred	Compiler1
Eugene	Database1
Sara	Database1
Sara	Database2
Eugene	Compiler1

DBProject

task
Database1
Database2

Completed/DBProject

Student
Fred
Sara

Generalized Projection

- → Extension of projection operation.
- → Allows arithmetic operations and string functions to be used in the projection list.

EXAMPLE

`employee(name, age, salary)`

Example: Find the name and monthly salary of all employees.

Answer:

$$\Pi_{name,salary/12}(employee)$$

AGGREGATION($\mathcal{G}$)

- Permits the use of aggregate functions (sum, average, min, max, count)
  - Take a collection of values and return a single value as a result.

$$\mathcal{G} \qquad (E)$$

$$G1, G2, \ldots Gn, \ldots F(A1), F(A2), \ldots Fm(Am)$$

Where,

E => any relational algebra expression

G1, G2, …..,Gn=> List of attributes on which to group.

F1,…Fm => Aggregate functions

A1, ….., A_m

EXAMPLE AGGREGATION($\mathcal{G}$)

instructor(id,name,dept_name,salary)
teaches(id, course_id, sec_id, semester,year)

1. Find out the sum of salaries of all instructors

Answer:

$$\mathcal{G}_{sum(salary)}(instructor)$$

2. Find the total no. of instructors who teach a course in the Spring 2010 semester.

Answer:

$$\mathcal{G}_{count-distinct(id)}(\sigma_{semester="spring"\wedge year=2010}(teaches))$$

3. Find out the average salary of all instructors

Answer:

$$\mathcal{G}_{average(salary)}(instructor)$$

4. Find out the average salary in each department

Answer:

$$dept\_name\mathcal{G}_{average(salary)}(instructor)$$

RELATIONAL CALCULUS

- Relational calculus is a non-procedural query language.
- It uses mathematical predicate calculus instead of algebra.
- It provides description about the query to get the result where as relational algebra gives the method to get the result.
- It informs the system what to do with the relation, but does not inform how to perform it. Relational calculus exists in two forms:
  - Tuple Relational Calculus
  - Domain Relational Calculus
- Relational calculus is mainly based on the well-known propositional calculus, which is a method of calculating with sentences or declarations. Such sentences or declarations, also termed as propositions, are ones for which a truth value(i.e. “True” or “False”) may be assigned.

TUPLE RELATIONAL CALCULUS

- Tuple relational calculus is used for selecting those tuples that satisfy the given condition. It is a logical language with variables ranging over tuples. In tuple relational calculus, we work on filtering tuples based on the given condition.

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

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

- In this form of relational calculus, we define tuple variable; specify the relation name in which the tuple is to be searched for, along with a condition. We can also specify column name using a dot operator, with the tuple variable to only get a certain attribute(column) in result.

- Example:

  {t | Employee(t) and t.Salary >10000} it selects the tuples from Employee relation such that resulting employee tuples will have salary greater than 10000.

  {t | Employees(t) and t.Dept_ID = 10} it selects all the tuples of employees name who work for Department 10.

DOMAIN RELATIONAL CALCULUS

- In contrast to tuple relational calculus, domain relational calculus uses list of attributes to be selected from the relation based on the condition. It is same as TRC but differs by selecting the attributes rather than selecting whole tuples. It is denoted as:
- {<a1, a2, a3, a4, ….., an> | P(a1, a2, a3,…, an)}
- Where a1, a2, a3, a4, ….., an are attributes of the relation and P is the condition.
- For example:

{<name,age> | <name,age> ∈ Student ^ age >17}

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

{<sid, name, address> | <sid, name, address> ∈ Student }

This query returns all records of student relation.

{<name> | <sid,name,address> ∈ Student^sid = 45}

This query gives the name of students whose id is 45.

Relational Algebra vs Relational Calculus

SN	Relational Algebra	Relational Calculus
1	Relational Algebra is a procedural language.	Relational Calculus is a declarative language.
2	Relational Algebra states how to obtain the result.	Relational Calculus states what result we have to obtain.
3	Relational Algebra describes the order in which operations have to be performed.	Relational calculus does not specify the order of operations.
4	Relational algebra is not domain dependent.	Relational Calculus can be domain dependent.
5	It is close to a programming language.	It is close to natural language.

EXAMPLES

Example 1: Consider following relational schema of a hospital where primary keys are underlined.

DOCTOR(<u>name</u>, age, address)

WORKS(name, deptno)

DEPARTMENT(<u>deptno</u>, floor, room)

Write down the relational algebra for the following:

1. List the room of the doctors named 'KAROL'

$$\prod_{room}\left(\sigma_{name="Karol"}(DEPARTMENT \bowtie WORKS \bowtie DOCTOR)\right)$$

2. Count the number of doctors working in top floor.

$$G_{count\ (name\ )}\left(\sigma_{Floor="top"}(DEPARTMENT \bowtie WORKS \bowtie DOCTOR)\right)$$

3. Delete all the department of ground floor

$$DEPARTMENT \leftarrow DEPARTMENT - (\sigma_{Floor="Ground"}(DEPARTMENT)$$

4. Insert a new department in the hospital.

   There if no hospital relation.

5. Increase the age of the doctor named 'KSROL' by 1.

$$\prod_{name\ ,age=age+1,address}\left(\sigma_{name="Ksrol"}(DOCTOR)\right)$$

Example 2: Consider following relational schema:

Student(StuName,StuId,Class,Major)

Course(CourseName,CourseNumber,CreditHours,Department)

Section(SectionId,CourseNumber,Semester,Year,Instructor-Name)

GradeReport(StuID, SectionID, Grade)

Prerequisite(CourseNumber,PrerequisiteNumber)

Write down the relational algebra for the following:

1. Display the name and Id of those students whose major is finance.

$$\prod_{StuName\ ,StuId}\left(\sigma_{Major\ ="Finance\ "}(Student)\right)$$

2. Display the number of courses taught by ‘Bhupi’

$$G_{count\ (Course\ Number\ )}\left(\sigma_{Instructor\ -Name\ ="Bhupi"}(Section)\right)$$

3. Change the credit hours of course whose course number equal to 11 to 4.

$$\prod_{CourseName\ ,CourseNumber\ ,CreditHours\ =4,Department} (\sigma_{CourseNumber\ =11}(Course))$$

4. Insert a new student {‘Raju’, 10,5,’DBMS’} in Student relation.

$$Student \leftarrow Student \cup \{"Raju", 10, 5, "DBMS"\}$$

5. Display the course name and course number of those courses whose credit hours is 3.

$$\prod_{CourseName\ ,CourseNumber} (\sigma_{CreditHours\ =3}(Course))$$

Thank You !!!