UNIVERSITY OF MINES AND TECHNOLOGY

ADVANCED DATABASE
CE 280

Presented BY: DR Eric affum

FORMAL QUERY LANGUAGES

What is FORMAL QUERY LANGUAGES?

- There are two kinds of query languages –
  - ❖ Relational algebra
  - ❖ Relational calculus

Relational Algebra

- ☆ It is a procedural query language.
- ☆ RA is a set of operations on relation(s).
- ☆ It is a set of algebraic operations.
- ☆ Input : One or more relations.
- ☆ Output : A relation.
- ☆ It provides a theoretical foundation for relational databases.
- ☆ Structured Query Language (SQL).
- ☆ It allows us to understand database operations in more detail and motivate us to write optimized queries.

Relational Algebra Operations

Fundamental operations:

- ☆ Select (σ)
- ☆ Project (Π)
- ☆ Union (U)
- ☆ Set Difference (-)
- ☆ Cartesian Product (X)
- ☆ Rename (ρ)

Relational Algebra Operations

Additional operations:

- ☆ Set intersection (∩)
- ☆ Assignment operation (=)
- ☆ Natural join ($\bowtie_*$)
- ☆ Division operation (÷)
- ☆ Outer join
  - Left outer join (⟕)
  - Right outer join (⟖)
  - Full outer join (⟗)

1. Select (σ)

- ☆ Select tuples that satisfy a given predicate.
- ☆ It is denoted by lowercase Greek letter sigma (σ).
- ☆ Syntax: $\sigma_{\text{<selection\_condition>}}$ (Relation).
- ☆ Example: $\sigma_{D\_ID = 2}$ (Employee).
- ☆ Comparison operators: =, ≠, <, ≤, >, and ≥.
- ☆ Connectives: AND (∧), OR (∨) and NOT (¬).

1. Select (σ)

Example 1: Write an RA expression to find all the instructors working in "Finance" department.

Solution:

$$\sigma_{\text{Dept\_Name} = \text{"Finance"}} (\text{INSTRUCTOR})$$

Output:

ID	Name	Dept_Name	Salary
26589	Yusuf	Finance	95000
12547	Neil	Finance	80000

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

1. Select (σ)

Example 2: Find all instructors with salary greater than $87,000.

Solution:

$$\sigma_{\text{Salary} > 87000}(\text{INSTRUCTOR})$$

Output:

ID	Name	Dept_Name	Salary
12121	Robin	Computer Science	90000
26589	Yusuf	Finance	95000
76985	Pratik	Computer Science	89000

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

1. Select (σ)

Example 3: Find all instructors who are working in "Finance" department and drawing the salary greater than $87,000.

Solution:

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

1. Select (σ)

Example 3: Find all instructors who are working in "Finance" department and drawing the salary greater than $87,000.

Solution:

$$\sigma_{\text{Dept\_Name} = \text{"Finance"} \wedge \text{Salary} > 87000} (\text{INSTRUCTOR})$$

Output:

ID	Name	Dept_Name	Salary
26589	Yusuf	Finance	95000

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

Homework

Question 1: Write an RA expression to find all the instructors drawing salary greater than 60000.

Question 2: Write an RA expression to find all the instructors drawing salary between 50000 and 75000.

Question 3: Write an RA expression to find all the instructors in the INSTRUCTOR relation.

2. Project (Π)

- ☆ It returns its argument relation with certain attributes left out
- ☆ It is a unary operator.
- ☆ It is denoted by the uppercase Greek letter pi (Π).
- ☆ Basically a relation is a set.
- ☆ In the result, the duplicate rows are eliminated.
- ☆ Syntax: $\Pi_{\text{Attribute1, Attribute2, ...}}$ (Relation).

2. Project (Π)

Example 1: List all instructors' ID, name, and salary, but do not care about the dept_name.

Solution:

$$\Pi_{\text{ID, Name, Salary}} (\text{INSTRUCTOR})$$

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

2. Project (Π)

Example 1: List all instructors' ID, name, and salary, but do not care about the dept_name.

Solution:

$$\Pi_{\text{ID, Name, Salary}} (\text{INSTRUCTOR})$$

Output:

ID	Name	Salary
10101	John	65000
12121	Robin	90000
25252	Alya	40000
26589	Yusuf	95000
54789	Ravi	60000
78787	Raj	87000
87458	Jayant	75000
76985	Pratik	89000
12547	Neil	80000

2. Project (Π)

Example 2: Find the name of all instructors in the Computer Science department.

Solution:

$$\Pi_{\text{Name}}(\sigma_{\text{Dept\_Name} = \text{"Computer Science"}}(\text{INSTRUCTOR}))$$

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

2. Project (Π)

Example 2: Find the name of all instructors in the Computer Science department.

Solution:

$$\Pi_{Name}(\sigma_{Dept\_Name = \text{"Computer Science"}}(INSTRUCTOR))$$

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

2. Project (Π)

Example 2: Find the name of all instructors in the Computer Science department.

Solution:

$$\Pi_{\text{Name}} (\sigma_{\text{Dept\_Name} = \text{"Computer Science"}} (\text{INSTRUCTOR}))$$

Output:

Name
Robin
Pratik

3. Union (U)

- ☆ Any relation is a set.
- ☆ Similar to union operation in Set Theory.
- ☆ It is a binary operator.
- ☆ It is a set of all objects that are a member of A, or B, or both.
- ☆ Like project, the duplicate rows are eliminated.
- ☆ It is denoted by U.
- ☆ Syntax: Π_{Column} (Relation_1) U Π_{Column} (Relation_2)

3. Union (U)

3. Union (U)

Example 1: List all customer names associated with the bank either as an account holder or a loan borrower.

3. Union (U)

Example 1: List all customer names associated with the bank either as an account holder or a loan borrower.

Solution: $\Pi_{Cu\_Name}(Depositor) \cup \Pi_{Cu\_Name}(Borrower)$

DEPOSITOR

R

Cu_Name	Acc_No
Tom	A-101
Amy	A-502
Rose	A-304
John	A-689

BORROWER

S

Cu_Name	Loan_No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

Cu_Name
Tom
Amy
Rose
John
Smith
Jack

3. Union (U)

Example 2: Find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both.

3. Union (U)

Example 2: Find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both.

Solution:

To find the set of all courses taught in the Fall 2009 semester, we write:

$$\Pi_{\text{Course\_id}} (\sigma_{\text{Semester} = \text{"Fall"} \wedge \text{year} = 2009} (\text{SECTION}))$$

To find the set of all courses taught in the Spring 2010 semester, we write:

$$\Pi_{\text{Course\_id}} (\sigma_{\text{Semester} = \text{"Spring"} \wedge \text{year} = 2010} (\text{SECTION}))$$

To answer the query, we need the union of these two sets:

$$\Pi_{\text{Course\_id}} (\sigma_{\text{Semester} = \text{"Fall"} \wedge \text{year} = 2009} (\text{SECTION})) \cup \Pi_{\text{Course\_id}} (\sigma_{\text{Semester} = \text{"Spring"} \wedge \text{year} = 2010} (\text{SECTION}))$$

Two important conditions

For R U S to be valid,

1. R and S must be of same arity.
2. For all i,

   Domain of the i^{th} attribute of R = Domain of i^{th} attribute of S.

Homework

Question 1: List all the course IDs which are taken in Spring 2020 or Fall 2021.

Question 2: List all the instructors' ID who taught courses in Spring 2020 or Fall 2021.

Instructor (ID, Name, Dept_Name, Salary)

Course (Course_ID, Title, Dept_Name, Credits)

Department (Dept_Name, Building, Budget)

Section (Course_ID, Sec_ID, Semester, Year, Building, Room_No, Time_slot_ID)

Teaches (ID, Course_ID, Sec_ID, Semester, Year)

Student (ID, Name, Dept_Name, Tot_Cred)

Advisor (S_ID, I_ID)

Takes (ID, Course_ID, Sec_ID, Semester, Year, Grade)

Classroom (Building, Room_Number, Capacity)

Time_Slot (Time_Slot_ID, Day, Start_Time, End_Time)

4. Set Difference (–)

- ☆ It is like the same set difference in Set Theory.
- ☆ It is a binary operation.
- ☆ To find tuples that are in one relation but are not in another.
- ☆ R - S = Tuples in R but not in S.

4. Set Difference (–)

4. Set Difference (–)

Example 1: List all customer names those who have a deposit account but not availed loan.

DEPOSITOR

Cu_Name	Acc_No
Tom	A-101
Amy	A-502
Rose	A-304
John	A-689

R

BORROWER

Cu_Name	Loan_No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

S

R - S

Cu_Name
Tom
Amy

Solution: $\Pi_{Cu\_Name}(Depositor) - \Pi_{Cu\_Name}(Borrower)$

Example – University Database

Instructor (ID, Name, Dept_Name, Salary)

Course (Course_ID, Title, Dept_Name, Credits)

Department (Dept_Name, Building, Budget)

Section (Course_ID, Sec_ID, Semester, Year, Building, Room_No, Time_slot_ID)

Teaches (ID, Course_ID, Sec_ID, Semester, Year)

Student (ID, Name, Dept_Name, Tot_Cred)

Advisor (S_ID, I_ID)

Takes (ID, Course_ID, Sec_ID, Semester, Year, Grade)

Classroom (Building, Room_Number, Capacity)

Time_Slot (Time_Slot_ID, Day, Start_Time, End_Time)

4. Set Difference (–)

Example 2: Find all the courses taught in the Fall 2009 semester but not in Spring 2010.

Solution:

To find the set of all courses taught in the Fall 2009 semester, we write:

4. Set Difference (–)

Example 2: Find all the courses taught in the Fall 2009 semester but not in Spring 2010.

Solution:

To find the set of all courses taught in the Fall 2009 semester, we write:

$$\Pi_{\text{Course\_id}} (\sigma_{\text{Semester} = \text{"Fall"} \wedge \text{year} = 2009} (\text{SECTION}))$$

To find the set of all courses taught in the Spring 2010 semester, we write:

$$\Pi_{\text{Course\_id}} (\sigma_{\text{Semester} = \text{"Spring"} \wedge \text{year} = 2010} (\text{SECTION}))$$

To answer the query, we need the minus of these two sets:

$$\Pi_{\text{Course\_id}} (\sigma_{\text{Semester} = \text{"Fall"} \wedge \text{year} = 2009} (\text{SECTION})) - \Pi_{\text{Course\_id}} (\sigma_{\text{Semester} = \text{"Spring"} \wedge \text{year} = 2010} (\text{SECTION}))$$

Two important conditions

For R - S to be valid,

1. R and S must be of same arity.
2. For all i,

   Domain of the i^{th} attribute of R = Domain of i^{th} attribute of S.

Homework1

R
Alphabets
A
B
C
D
E
F

S
Characters
A
B
$
L
Y
1
#

S - R

S - R

Homework2

Question 1: List all the instructors' ID who are not advisors.

Question 2: List all the instructors' ID who taught in Spring 2020 but not advisors.

Instructor (ID, Name, Dept_Name, Salary)

Course (Course_ID, Title, Dept_Name, Credits)

Department (Dept_Name, Building, Budget)

Section (Course_ID, Sec_ID, Semester, Year, Building, Room_No, Time_slot_ID)

Teaches (ID, Course_ID, Sec_ID, Semester, Year)

Student (ID, Name, Dept_Name, Tot_Cred)

Advisor (S_ID, I_ID)

Takes (ID, Course_ID, Sec_ID, Semester, Year, Grade)

Classroom (Building, Room_Number, Capacity)

Time_Slot (Time_Slot_ID, Day, Start_Time, End_Time)

5. Cartesian Product (X)

- ☆ Cartesian product associates every tuple of R_1 with every tuple of R_2.
- ☆ It is a binary operation.
- ☆ It is denoted by cross (X) symbol.
- ☆ R_1 X R_2 = All possible pairing.
- ☆ Same attribute may appear in R_1 and R_2.
- ☆ R = Depositor X Borrower.

5. Cartesian Product (X)

Example: Perform Depositor X Borrower.

DEPOSITOR

Cu_Name	Acc_No
Tom	A-101
Rose	A-304

BORROWER

Cu_Name	Loan_No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

5. Cartesian Product (X)

Example: Perform Depositor X Borrower.

DEPOSITOR

Cu_Name	Acc_No
Tom	A-101
Rose	A-304

BORROWER

Cu_Name	Loan_No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

DEPOSITOR X BORROWER

Cu_Name	Acc_No	Cu_Name	Loan_No
Tom	A-101	John	L-201
Tom	A-101	Smith	L-658
Tom	A-101	Rose	L-254
Tom	A-101	Jack	L-547
Rose	A-304	John	L-201
Rose	A-304	Smith	L-658
Rose	A-304	Rose	L-254
Rose	A-304	Jack	L-547

5. Cartesian Product (X)

- ☆ R = Depositor X Borrower.
- ☆ R = (Depositor.Cu_Name, Depositor.Acc_No, Borrower.Cu_Name, Borrower.Loan_No).
- ☆ R = (Depositor.Cu_Name, Acc_No, Borrower.Cu_Name, Loan_No).

Example – University Database

Instructor (ID, Name, Dept_Name, Salary)

Course (Course_ID, Title, Dept_Name, Credits)

Department (Dept_Name, Building, Budget)

Section (Course_ID, Sec_ID, Semester, Year, Building, Room_No, Time_slot_ID)

Teaches (ID, Course_ID, Sec_ID, Semester, Year)

Student (ID, Name, Dept_Name, Tot_Cred)

Advisor (S_ID, I_ID)

Takes (ID, Course_ID, Sec_ID, Semester, Year, Grade)

Classroom (Building, Room_Number, Capacity)

Time_Slot (Time_Slot_ID, Day, Start_Time, End_Time)

5. Cartesian Product (X)

Example : Find the names of all instructors in the Physics department together with the course id of all courses they taught.

Solution:

$\sigma_{\text{Dept\_Name = "Physics"}}$ (INSTRUCTOR X TEACHES)

$\sigma_{\text{Instructor.ID = Teaches.ID}}$ ($\sigma_{\text{Dept\_Name = "Physics"}}$ (INSTRUCTOR X TEACHES))

$\Pi_{\text{name, course\_id}}$ ($\sigma_{\text{Instructor.ID = Teaches.ID}}$ ($\sigma_{\text{Dept\_Name = "Physics"}}$ (INSTRUCTOR X TEACHES)))

5. Cartesian Product (X)

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
78787	Raj	Physics	87000

TEACHES

ID	Course_ID	Sec_ID	Semester	Year
10101	BIO-108	1	Summer	2009
20202	CS-103	1	Spring	2010
78787	PHY-101	2	Fall	2011
12345	EE-203	1	Spring	2009

5. Cartesian Product (X)

INSTRUCTOR X TEACHES

INSTRUCTOR.ID	Name	Dept_Name	Salary	TEACHES.ID	Course_ID	Sec_ID	Semester	Year
10101	John	Biology	65000	10101	BIO-108	1	Summer	2009
10101	John	Biology	65000	20202	CS-103	1	Spring	2010
10101	John	Biology	65000	78787	PHY-101	2	Fall	2011
10101	John	Biology	65000	12345	EE-203	1	Spring	2009
78787	Raj	Physics	87000	10101	BIO-108	1	Summer	2009
78787	Raj	Physics	87000	20202	CS-103	1	Spring	2010
78787	Raj	Physics	87000	78787	PHY-101	2	Fall	2011
78787	Raj	Physics	87000	12345	EE-203	1	Spring	2009

5. Cartesian Product (X)

$\sigma_{Dept\_Name = "Physics"}$ (INSTRUCTOR X TEACHES)

INSTRUCTOR.ID	Name	Dept_Name	Salary	TEACHES.ID	Course_ID	Sec_ID	Semester	Year
10101	John	Biology	65000	10101	BIO-108	1	Summer	2009
10101	John	Biology	65000	20202	CS-103	1	Spring	2010
10101	John	Biology	65000	78787	PHY-101	2	Fall	2011
10101	John	Biology	65000	12345	EE-203	1	Spring	2009
78787	Raj	Physics	87000	10101	BIO-108	1	Summer	2009
78787	Raj	Physics	87000	20202	CS-103	1	Spring	2010
78787	Raj	Physics	87000	78787	PHY-101	2	Fall	2011
78787	Raj	Physics	87000	12345	EE-203	1	Spring	2009

5. Cartesian Product (X)

$\sigma_{Dept\_Name = "Physics"}$ (INSTRUCTOR X TEACHES)

INSTRUCTOR.ID	Name	Dept_Name	Salary	TEACHES.ID	Course_ID	Sec_ID	Semester	Year
78787	Raj	Physics	87000	10101	BIO-108	1	Summer	2009
78787	Raj	Physics	87000	20202	CS-103	1	Spring	2010
78787	Raj	Physics	87000	78787	PHY-101	2	Fall	2011
78787	Raj	Physics	87000	12345	EE-203	1	Spring	2009

5. Cartesian Product (X)

$$\sigma_{\text{Instructor.ID} = \text{Teaches.ID}} (\sigma_{\text{Dept\_Name} = \text{"Physics"}} (\text{INSTRUCTOR X TEACHES}))$$

INSTRUCTOR.ID	Name	Dept_Name	Salary	TEACHES.ID	Course_ID	Sec_ID	Semester	Year
78787	Raj	Physics	87000	10101	BIO-108	1	Summer	2009
78787	Raj	Physics	87000	20202	CS-103	1	Spring	2010
78787	**Raj**	**Physics**	**87000**	**78787**	**PHY-101**	**2**	**Fall**	**2011**
78787	Raj	Physics	87000	12345	EE-203	1	Spring	2009

5. Cartesian Product (X)

$\sigma_{Instructor.ID = Teaches.ID} (\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES))$

INSTRUCTOR.ID	Name	Dept_Name	Salary	TEACHES.ID	Course_ID	Sec_ID	Semester	Year
78787	Raj	Physics	87000	78787	PHY-101	2	Fall	2011

5. Cartesian Product (X)

$$\sigma_{\text{Instructor.ID} = \text{Teaches.ID}} (\sigma_{\text{Dept\_Name} = \text{"Physics"}} (\text{INSTRUCTOR X TEACHES}))$$

INSTRUCTOR.ID	Name	Dept_Name	Salary	TEACHES.ID	Course_ID	Sec_ID	Semest
78787	Raj	Physics	87000	78787	PHY-101	2	Fall

$$\Pi_{\text{name, course\_id}} (\sigma_{\text{Instructor.ID} = \text{Teaches.ID}} (\sigma_{\text{Dept\_Name} = \text{"Physics"}} (\text{INSTRUCTOR X TEACH}$$

Name	Course_ID
Raj	PHY-101

5. Cartesian Product (X)

Example : Find the names of all instructors in the Physics department together with the course id of all courses they taught.

Solution:

$\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES)$

$\sigma_{Instructor.ID = Teaches.ID} (\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES))$

$\Pi_{name, course\_id} (\sigma_{Instructor.ID = Teaches.ID} (\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES)))$

5. Cartesian Product (X)

Example : Find the names of all instructors in the Physics department together with the course id of all courses they taught.

Solution:

$\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES)$

$\sigma_{Instructor.ID = Teaches.ID} (\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES))$

$\Pi_{name,\ course\_id} (\sigma_{Instructor.ID = Teaches.ID} (\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES)))$

[or]

$\Pi_{name,\ course\_id} (\sigma_{Instructor.ID = Teaches.ID} ((\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR) \times TEACHES))$

6. Rename (ρ)

- ☆ Relations in the database have names.
- ☆ The results of relational-algebra expressions do not have a name.
- ☆ It is useful to be able to give them names.
- ☆ It is a unary operation.
- ☆ It is denoted by the lowercase Greek letter rho (ρ).
- ☆ Syntax: ρ_x (E)

6. Rename (ρ)

☆ A relation *r* by itself is considered a trivial relational-algebra expression.

'r' is not a relational algebra expression, rather 'r' is the name of the relation. However, with rename operator it is considered as a trivial relational algebra expression.

6. Rename (ρ)

- ☆ A relation *r* by itself is considered a trivial relational-algebra expression.
- ☆ Thus, the same rename operation can be applied to a relation *r* to get the same relation under a new name.
- ☆ The results of relational-algebra expressions do not have a name.
- ☆ $\rho_{x(A1,A2,\ldots An)}$ (Expression)

6. Rename (ρ)

Example: Perform Depositor X Borrower.

DEPOSITOR

Cu_Name	Acc_No
Tom	A-101
Rose	A-304

BORROWER

Cu_Name	Loan_No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

DEPOSITOR X BORROWER

Cu_Name	Acc_No	Cu_Name	Loan_No
Tom	A-101	John	L-201
Tom	A-101	Smith	L-658
Tom	A-101	Rose	L-254
Tom	A-101	Jack	L-547
Rose	A-304	John	L-201
Rose	A-304	Smith	L-658
Rose	A-304	Rose	L-254
Rose	A-304	Jack	L-547

6. Rename (ρ)

Example : Find the names of all instructors in the Physics department together with the course id of all courses they taught and name the resultant relation as Ins_Phy.

6. Rename (ρ)

Example : Find the names of all instructors in the Physics department together with the course id of all courses they taught and name the resultant relation as Ins_Phy.

Solution:

$$\sigma_{\text{Dept\_Name = "Physics"}} (\text{INSTRUCTOR X TEACHES})$$

$$\sigma_{\text{Instructor.ID = Teaches.ID}} (\sigma_{\text{Dept\_Name = "Physics"}} (\text{INSTRUCTOR X TEACHES}))$$

$$\Pi_{\text{name, course\_id}} (\sigma_{\text{Instructor.ID = Teaches.ID}} (\sigma_{\text{Dept\_Name = "Physics"}} (\text{INSTRUCTOR X TEACHES})))$$

[or]

$$\Pi_{\text{name, course\_id}} (\sigma_{\text{Instructor.ID = Teaches.ID}} ((\sigma_{\text{Dept\_Name = "Physics"}} (\text{INSTRUCTOR}) \text{ X TEACHES}))$$

6. Rename (ρ)

Example : Find the names of all instructors in the Physics department together with the course id of all courses they taught and name the resultant relation as Ins_Phy.

Solution:

$\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES)$

$\sigma_{Instructor.ID = Teaches.ID} (\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES))$

$\Pi_{name, course\_id} (\sigma_{Instructor.ID = Teaches.ID} (\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR \times TEACHES)))$

[or]

$\Pi_{name, course\_id} (\sigma_{Instructor.ID = Teaches.ID} ((\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR) \times TEACHES))$

$\rho_{Ins\_Phy} (\Pi_{name, course\_id} (\sigma_{Instructor.ID = Teaches.ID} ((\sigma_{Dept\_Name = "Physics"} (INSTRUCTOR) \times TEACHES)))$

Relational Calculus

- Relational calculus is a non-procedural query language (or declarative language)
- It uses mathematical predicate calculus (or first-order logic) instead of algebra
- RC tells what to do but never explain how to do it
- RC provides description about the query to get the result where as
- Relational algebra gives the method to get the result

Relational Calculus

- When to apply to database, it comes in two flavors

1. Tuple Relational Calculus (TRC)
   - Proposed by Codd in the year 1972
   - Works on tuples (or rows)

- 2. Domain Relational Calculus (DRC)
   - Proposed by Locroix & Pirotte in the year 1977
   - Works on domain of attributes (or Columns)

Relational Calculus

- Calculus has variables, constants, comparison operators, logical connectives and quantifiers.
- TRC: variables ranges over tuples
  - Like SQL
- DRC : Variables range over domain elements
  - Like Query-By-Example (QBE)
- Expression in calculus are called formulas
- Resulting tuples is an assignment of constant to variable that make the formula evaluate to true

Tuple Relational Calculus

Result of the query:
It is the set of all tuples t such that predicate P is true for t

- ★ A nonprocedural query language.
- ★ Query: $\{t \mid P(t)\}$.
- ★ Free variable unless it is quantified by a $\exists$ or $\forall$.
- ★ $t \in \text{instructor} \wedge \exists s \in \text{department}(t[\text{dept\_name}] = s[\text{dept\_name}])$.
- ★ A TRC formula is built up out of atoms.

  → $s \in r$

  → $s[x]$ $(<, \leq, =, \neq, >, \geq)$ $u[y]$

  → $s[x]$ $(<, \leq, =, \neq, >, \geq)$ c

Tuple Relational Calculus

Formula:

1. Set of domain variable and constant
2. Set of comparison operator e.g. $<, \leq, =, \neq, >, \geq$
3. Set of connectives $\wedge$, $\vee$, $\neg$
4. Implication ($\Rightarrow$) $(x \Rightarrow y)$ if x is true, then y is true $(x \Rightarrow y \equiv \neg x \vee y)$
5. Quantifiers : Existential Quantifiers ($\exists$) and Universal Quantifier ($\forall$)

   $\exists\, x\,\big(P(x)\big)\ and\ \forall x(P(x))$ x is free domain variable

Tuple Relational Calculus

- $\exists\ t \in r\ (Q(t)) \equiv$ "there exist" a tuple in t in relation r such that Q(t) is true
- $\forall t\ \in r(Q(t)) \equiv Q$ is true for all tuples t in relation r

➢ Free code bound variable
  - The use of quantifiers $\exists x\ and\ \forall x$ in a formula is said to bind x in the formula
  - A variable that is not bound is free
  - Lets revisits the definition of query: {t|P(t)}
  - There are important restrictions

-The variable t that appears to the left of '|' must be the only variable in the formula P(t).

-In other words, all other tuple variables must be bound using a quantifier

Tuple Relational Calculus

- ★ An atom is a formula.
- ★ If P_1 is a formula, then so are $\neg P_1$ and (P_1).
- ★ If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$, and $P_1 \Rightarrow P_2$.
- ★ If $P_1(s)$ is a formula containing a free tuple variable s, and r is a relation, then $\exists s \in r\ (P_1(s))$ and $\forall s \in r\ (P_1(s))$.
- ★ $P_1 \wedge P_2$ is equivalent to $\neg(\neg(P_1) \vee \neg(P_2))$.
- ★ $\forall t \in r\ (P_1(t))$ is equivalent to $\neg\exists t \in r\ (\neg P_1(t))$.
- ★ $P_1 \Rightarrow P_2$ is equivalent to $\neg(P_1) \vee P_2$.

Safety of Expressions

- ★ $\{t \mid \neg(t \in \text{instructor})\}$.
- ★ Unsafe expressions.
- ★ Domain of a tuple relational formula.
- ★ dom(P).
- ★ $\text{dom}(t \in \text{instructor} \wedge t[\text{salary}] > 80000)$.
- ★ $\{t \mid P(t)\}$ is safe if all values in the result are from dom(P).

Tuple Relational Calculus

★ A nonprocedural query language.

★ Query: {t | P(t)}.

ID	Name	Dept_Name	Salary
1001	John Abraham	QA	85000
1002	Kinglsey David	Testing	91000
1003	Naveen Kumar	QA	84000
1004	Rishi Kumar	HR	25000
1005	Mohamad Rafi	Testing	92000
1006	Suresh Raj	HR	24000
1007	Ram Kumar	Admin	15000

Example Queries

Example 1: Find the ID, name, dept name, salary for instructors whose salary is greater than $80,000:

$$\{t \mid t \in \text{instructor} \wedge t[\text{salary}] > 80000\}$$

Tuple Relational Calculus

- ★ A nonprocedural query language.
- ★ Query: $\{t \mid P(t)\}$.
- ★ Free variable unless it is quantified by a $\exists$ or $\forall$.
- ★ $t \in \text{instructor} \wedge \exists s \in \text{department}(t[\text{dept\_name}] = s[\text{dept\_name}])$.
- ★ A TRC formula is built up out of atoms.

  → $s \in r$

  → $s[x]$ $(<, \leq, =, \neq, >, \geq)$ $u[y]$

  → $s[x]$ $(<, \leq, =, \neq, >, \geq)$ c

Example Queries

Example 1: Find the ID, name, dept name, salary for instructors whose salary is greater than $80,000:

$\{t \mid t \in \text{instructor} \wedge t[\text{salary}] > 80000\}$

Example 2: Find the instructor ID for each instructor with a salary greater than $80,000:

$\exists t \in r\ (Q(t))$

$\{t \mid \exists s \in \text{instructor}\ (t[\text{ID}] = s[\text{ID}] \wedge s[\text{salary}] > 80000)\}$

Example Queries

Example 3: Find the names of all instructors whose department is in the Watson building:

$$\{t \mid \exists s \in \text{instructor}\ (t[\text{name}] = s[\text{name}] \wedge \exists u \in \text{department}\ (u[\text{dept name}] = s[\text{dept name}] \wedge u[\text{building}] = \text{“Watson”}))\}$$

Example Queries

Example 4: Find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both

$$\{t \mid \exists s \in section\ (t[course\ id\ ] = s[course\ id\ ]) \wedge s[semester] = \text{“Fall”} \wedge s[year] = 2009)\} \vee$$

$$\exists u \in section\ (u[course\ id\ ] = t[course\ id\ ])\} \wedge u[semester] = \text{“Spring”} \wedge u[year] = 2010)\}$$

Example Queries

Example 4: Find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both

$$\{t \mid \exists s \in section\ (t[course\ id\ ] = s[course\ id\ ]) \wedge s[semester] = \text{“Fall”} \wedge s[year] = 2009)\} \vee$$
$$\exists u \in section\ (u[course\ id\ ] = t[course\ id\ ])\} \wedge u[semester] = \text{“Spring”} \wedge u[year] = 2010)\}$$

Example 5: Find only those course id values for courses that are offered in both the Fall 2009 and Spring 2010 semesters

$$\{t \mid \exists s \in section\ (t[course\ id\ ] = s[course\ id\ ]) \wedge s[semester] = \text{“Fall”} \wedge s[year] = 2009)\} \wedge$$
$$\exists u \in section\ (u[course\ id\ ] = t[course\ id\ ])\} \wedge u[semester] = \text{“Spring”} \wedge u[year] = 2010)\}$$

Example Queries

Example 5: Find only those course id values for courses that are offered in both the Fall 2009 and Spring 2010 semesters

Example 6: Find all the courses taught in the Fall 2009 semester but not in Spring 2010 semester

Domain Relational Calculus

- Domain Relational Calculus is a non-procedural query language
- In DRC the records are filtered based on the domains
- DRC uses list of attributes to be selected from relation based on condition (or predicate)
- DRC is same as TRC but differs by selecting attributes rather than selecting whole tuples
- In DRC , Each query is an expression of the form:
- $\{< a_1, a_2 \ \ldots a_n > | P(a_1, a_2 \ldots a_n)\}$
- $a_1, a_2 \ \ldots a_n$, represent domain variables
- P represent predicate similar to that of the predicate calculus
- Results of Query: it's the set of all tuples $a_1, a_2 \ \ldots a_n$ such that predicate P is true for $a_1, a_2 \ \ldots a_n$ tuples
-

Predicate Calculus Formula

- **Notations used:**

➢ $< a_1, a_2 \ \ldots a_n > \in r$, where r is on attributes and $a_1, a_2 \ \ldots a_n$ are domain variables or domain constant

➢ P is a formula similar to that of the predicate calculus

Formula:

1. Set of domain variable and constant
2. Set of comparison operator e.g. $<, \leq, =, \neq, >, \geq$
3. Set of connectives and, or, not
4. Implication ($\Rightarrow$) $(x \Rightarrow y)$ if x is true, then y is true $(x \Rightarrow y \equiv \neg x \vee y)$
5. Quantifiers : Existential Quantifiers ($\exists$) and Universal Quantifier ($\forall$)

$$\exists\, x \left(P(x)\right) and\ \forall x (P(x))$$

x is free domain variable

Example Queries: DRC

- Find the loan-number, branch-name, and amount for loans of over $1200

$$\{< l, b, a > \mid < l, b, a > \in loan \wedge a > 1200\}$$

Find the loan-number for each loan of an amount greater than $1200

$$\{ t \mid \exists s \in loan \ (t[loan\text{-}number] = s[loan\text{-}number] \wedge s[amount] > 1200)\}$$

branch (branch-name, branch-city, assets)
customer (customer-name, customer-street, customer-city)
account (account-number, branch-name, balance)
loan (loan-number, branch-name, amount)
depositor (customer-name, account-number)
borrower (customer-name, loan-number)

Example Queries: DRC

- Find the loan of all customers who have a loan of over $1200

$$\{< c > \mid \exists\, l, b, a\, (< c, l > \in borrower \wedge (< l, b, a > \in loan \wedge a > 1200))\}$$

branch (branch-name, branch-city, assets)
customer (customer-name, customer-street, customer-city)
account (account-number, branch-name, balance)
loan (loan-number, branch-name, amount)
depositor (customer-name, account-number)
borrower (customer-name, loan-number)

Home Work: DRC

- 1. Find the names of all customers who have a loan of over $1200

$$\{< c > \mid \exists\, l, b, a\, (< c, l > \in borrower \wedge (< l, b, a > \in loan \wedge a > 1200))\}$$

- 2. Find the name of all customers having a loan at the Perryridge branch and find the loan amount

branch (branch-nam)e, branch-city, assets)
customer (customer-name, customer-street, customer-city)
account (account-number, branch-name, balance)
loan (loan-number, branch-name, amount)
depositor (customer-name, account-number)
borrower (customer-name, loan-number)

Home Work: DRC

- 3. Find the name oof all customers having a loan, an amount or both at the bank

branch (branch-nam)e, branch-city, assets)
customer (customer-name, customer-street, customer-city)
account (account-number, branch-name, balance)
loan (loan-number, branch-name, amount)
depositor (customer-name, account-number)
borrower (customer-name, loan-number)