Topics Covered in Todays Class

Unit 2:
Introduction to Relational Algebra

Why you should Learn “Relational Algebra” ?

Why you should Learn “Relational Algebra” ?

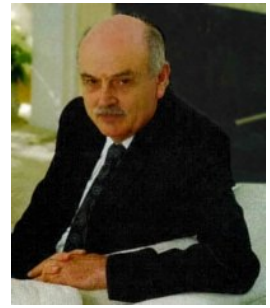
Relational Algebra is

- **Core of** Relational Query Language, Example: **SQL**
- Provides framework for Query implementation and optimization
- Is a ***mathematical language*** for manipulating relations.

Relational Algebra based on Relational Model

student_info

USN	Name
1BM14CS001	Arjun
1BM14CS002	Balaji



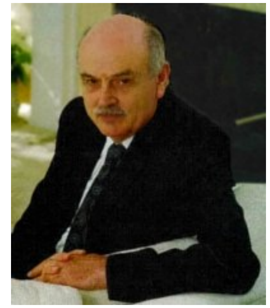
Won Turing award 1981

Relational model due to Edgar Teds Codd, a mathematician at IBM in 1970

Relational Algebra based on Relational Model

student_info

USN	Name
1BM14CS001	Arjun
1BM14CS002	Balaji



Won Turing award 1981

Relational model due to Edgar "Ted" Codd, a mathematician at IBM in 1970

Column or Attribute

Row or Tuple or Record

USN	Name
1BM14CS001	Arjun
1BM14CS002	Balaji

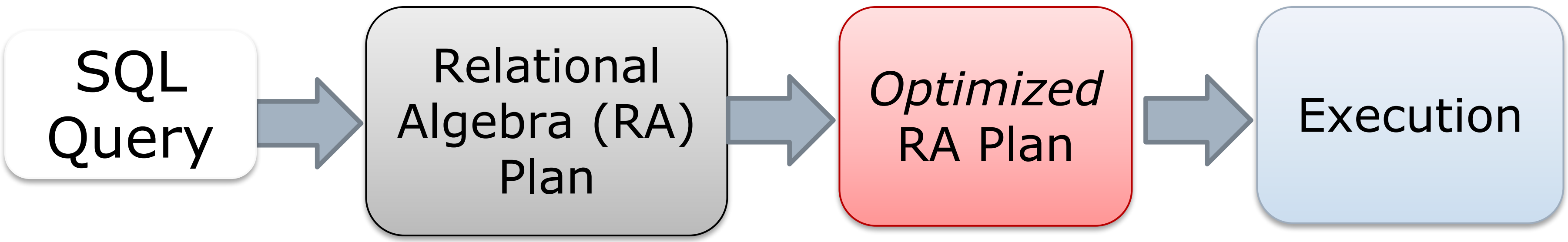
Table or Relation

Database table Name: student_info

RDBMS Architecture

Relational Database Management System (RDBMS)

How does a SQL engine work ?



Declarative query (from user)

Translate to relational algebra expression

Find logically equivalent- but **more efficient**- RA expression

Execute each operator of the **optimized** plan!

RDBMS Architecture

Relational Database Management System (RDBMS)

How does a SQL engine work ?

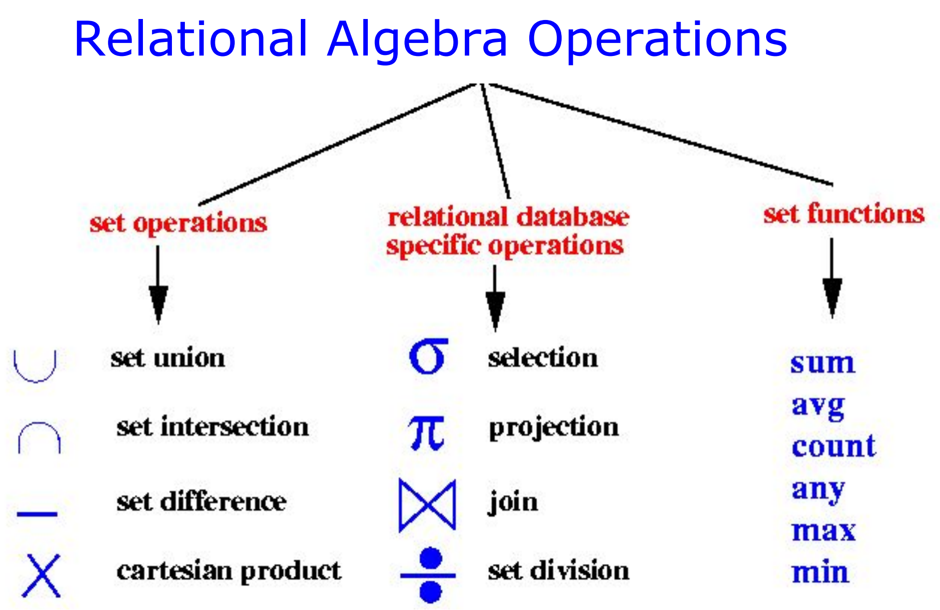
Relational Algebra allows us to translate declarative (SQL) queries into precise and optimizable expressions!

What is an “Algebra”

- Mathematical system consisting of:
  - Operands: variables or values from which new values can be constructed.
  - Operators: symbols denoting procedures that construct new values from given values.

What is Relational Algebra (RA) ?

- An algebra whose **operands** are database **tables** or **relations** or variables that represent relations.
- **Operators** are designed to do the most common things that we need to do with relations in a database.
  - The result is an algebra that can be used as a query language for relations.



Keep in mind: RA operates on sets!

- RDBMSs use *multisets*, however in relational algebra formalism we will consider **sets!**

- Also: we will consider the ***named perspective***, where every attribute must have a unique name
  - ☐attribute order does not matter…

Now on to the basic RA operators…

Unary Relational Operations

- Select Operation (σ sigma)
- Project Operation (Π phi)

Select Operation (σ sigma)

- It selects tuples that satisfy the given predicate from a relation.
- Select written as

Notation $\sigma_{\textit{selection condition}}(R)$

- Where **σ** stands for selection predicate
- **R** stands for relation / table name
- *Selection condition* is prepositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like − $=, \neq, \geq, <, >, \leq$.

Select Operation (σ)

Select Operation (σ)
Returns all tuples which satisfy a condition

$\sigma_{\textit{selection condition}}(R)$

SQL:

```
SELECT *
FROM Student
WHERE marks > 60;
```

Equivalent Relational Algebra Expression

Student

usn	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

usn	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14IS001	Avinash	20	90

Select Operation (σ)

Select Operation (σ)
Returns all tuples which satisfy a condition

$\sigma_{selection\ condition}(R)$

SQL:

```
SELECT *
FROM Student
WHERE marks > 60;
```

Student

usn	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

Relational Algebra Expression

$\sigma_{\text{marks} > 60}(\text{Student})$

usn	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14IS001	Avinash	20	90

Select Operation (σ)

Select Operation (σ)
Returns all tuples which satisfy a condition

$$\sigma_{\textit{selection condition}}(R)$$

SQL:

```
SELECT *
FROM Student
WHERE name='Avinash';
```

Student

usn	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

Relational Algebra Expression

$$\sigma_{name='Avinash'}(Student)$$

usn	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14IS001	Avinash	20	90

Project Operation (Π phi)

- It projects column(s) that satisfy a given predicate.
- Written as

Notation $\Pi_{A1,....,An}(R)$

- Where A_1, A_2 , A_n are attribute names of relation **R**.
- **Duplicate rows are automatically eliminated**, as relation is a set.

Project Operation (Π phi)

- Eliminates columns, then removes duplicates
- Written as

Notation $\Pi_{A1,\ldots,An}(R)$

SQL:

```
SELECT DISTINCT
 dep_num
FROM Student;
```

Relational Algebra Expression

$\Pi_{dep\_num}(Student)$

dep_num
10
20

Student

usn	name	dep_num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

Project Operation (Π phi)

- Eliminates columns, then removes duplicates
- Written as

Notation $\Pi_{A1,\ldots,An}(R)$

SQL:

```
SELECT DISTINCT
 name, dep_num
FROM Student;
```

Relational Algebra Expression

$\Pi_{name,\ dep\_num}(Student)$

Student

usn	name	dep_num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

name	dep_num
Avinash	10
Balaji	10
Chandan	10
Dinesh	10
Avinash	20

Note that RA Operators are Compositional!

Student

usn	name	dep_num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

name	marks
Avinash	100
Balaji	80
Avinash	90

```
SELECT DISTINCT
name, marks
FROM Student
WHERE marks > 60;
```

How do we represent this query in Relational Algebra?

Note that RA Operators are Compositional!

Student

<u>usn</u>	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

name	marks
Avinash	100
Balaji	80
Avinash	90

```
SELECT DISTINCT
name,  marks
FROM Student
WHERE marks > 60;
```

$$\Pi_{name,marks}(\sigma_{marks>60}(Student))$$

How do we represent this query in Relational Algebra?

Note that RA Operators are Compositional!

Student

usn	name	dep_num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

name	marks
Avinash	100
Balaji	80
Avinash	90

```
SELECT DISTINCT
name, marks
FROM Student
WHERE marks > 60;
```

$$\Pi_{name,marks}(\sigma_{marks>60}(Student))$$

$$\sigma_{marks>60}(\Pi_{name,marks}(Student))$$

How do we represent this query in Relational Algebra?

Are these logically equivalent?

Note that RA Operators are Compositional!

Student

<u>usn</u>	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

name	marks
Avinash	100
Balaji	80
Avinash	90

```
SELECT DISTINCT
name, marks
FROM Student
WHERE marks > 60;
```

$$\Pi_{name,marks}(\sigma_{marks>60}(Student))$$

$$\updownarrow$$

$$\sigma_{marks>60}(\Pi_{name,marks}(Student))$$

How do we represent this query in Relational Algebra?

Are these logically equivalent?

Question

Student

<u>usn</u>	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	100

What is the output of following relational algebra query ?

```
SELECT DISTINCT
name,  marks
FROM Student
WHERE marks > 60;
```

$$\Pi_{name,marks}(\sigma_{marks>60}(Student))$$

Question

Student

<u>usn</u>	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	100

What is the output of following relational algebra query ?

```
SELECT DISTINCT
name,  marks
FROM Student
WHERE marks > 60;
```

$$\Pi_{name,marks}(\sigma_{marks>60}(Student))$$

name	marks
Avinash	100
Balaji	80

Question

Student

<u>usn</u>	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	100

What is the output of following relational algebra query ?

```
SELECT DISTINCT
usn, name, marks
FROM Student
WHERE marks > 60;
```

$$\Pi_{usn,name,marks}(\sigma_{marks>60}(Student))$$

Question

Student

usn	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	100

What is the output of following relational algebra query ?

```
SELECT DISTINCT
usn, name,  marks
FROM Student
WHERE marks > 60;
```

$$\Pi_{usn,name,marks}(\sigma_{marks>60}(Student))$$

usn	name	marks
1BM14CS001	Avinash	100
1BM14CS002	Balaji	80
1BM14IS001	Avinash	100

Question

- Consider the relational schema Schedule containing Theater name, Movie Title and timing at which movie will be played. Time in the Schedule table is stored in 24Hr format i.e., 6:00pm will be stored as 18:00

Schedule(Theater, MovieTitle, Time)

Write the following query both in Relational Algebra and SQL.

List the theater name and time where we can watch the movie "FndingNemo" after 3pm.

Question

- Consider the relational schema Schedule containing Theater name, Movie Title and timing at which movie will be played. Time in the Schedule table is stored in 24Hr format i.e., 6:00pm will be stored as 18:00

Schedule(Theater, MovieTitle, Time)

Write the following query both in Relational Algebra and SQL.

List the theater name and time where we can watch the movie "FndingNemo" after 3pm.

```
SELECT Theater, Time
FROM Schedule
WHERE MovieTitle = 'FindingNemo' AND time >= 15:00;
```

$$\Pi_{\text{Theater,Time}}(\sigma_{\text{MovieTitle='FindingNemo' } \mathbf{AND} \text{ Time} \geq \ 15:00}(\text{Schedule}))$$

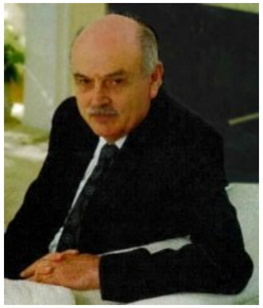
Database Philosophy

God made the integers;
all else is the work of man.

(Leopold Kronecker, 19th Century Mathematician)

Codd made relations;
all else is the work of man.

(Raghu Ramakrishnan, DB text book author)



Edgar Ted Codd

Won Turing award 1981

Rename Operation (ρ rho)

- The rename operation allows us to rename the either relation name or attribute names or both.
- Written as

Notation

$$\rho_{S(B1,\ldots,\ Bn)}\ (R) \text{ or } \rho_S(R) \text{ or } \rho_{(B1,\ldots,\ Bn)}\ (R)$$

- S is the new relation name and B_1, B_2....B_n are new attribute names.

Rename Operation (ρ rho)

- The rename operation allows us to rename the either relation name or attribute names or both.

Student

usn	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	Avinash	20	90

SQL:

```
SELECT
usn as Student_USN
name as Student_Name
dep_num as Department_Number
FROM Student;
```

$$\rho_{\text{Student\_USN, Student\_Name, Department\_Number}}(\text{Student})$$

Student

Student_USN	Student_Name	Department_Number
1BM14CS001	Avinash	10
1BM14CS002	Balaji	10
1BM14CS003	Chandan	10
1BM14CS004	Dinesh	10
1BM14IS001	Avinash	20

Binary Operations

Cartesian Product or Cross-Product or Cross Join (×)

- The **CARTESIAN PRODUCT** or **CROSS JOIN** returns the Cartesian product of the sets of records from the two or more joined tables.
- Notation: table1 × table2

table1

ID	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

```
select * from table1,table2;
```

Relational Algebra Expression

table1 × table2

ID	M	ID	N
1	a	2	p
1	a	3	q
1	a	5	r
2	b	2	p
2	b	3	q
2	b	5	r
4	c	2	p
4	c	3	q
4	c	5	r

Join Operation (⋈)

Join Operation

Written as $R \bowtie_{joincondition} S$

Emp_Dept

Emp_ID	D_ID
121	2
240	4

Dept

D_num	D_name
1	Sale
2	Account
4	Marketing

$Emp\_Dept \bowtie_{D\_ID=D\_num} Dept$

Emp_ID	D_ID	D_num	D_name
121	2	2	Account
240	4	4	Marketing

Join Operation (⋈)

- Join Written as R $\bowtie_{joincondition}$ S
- JOIN operation is used to combine related records from two tables into a single records
- A general join condition is of the form

  <condition> AND <condition> AND……AND <condition>
- Where each condition is of the form A_i θ B_j , A_i is an attribute in relation R and B_j is an attribute in relation S, Ai and Bj have same domain
  - It is called a ***theta join***, if θ is one of the comparison operators {**< , > , = , ≠ , ≤ , ≥**}.
  - Once again, if θ is **=** it's called an **equijoin**, and
  - if the equijoin is on same-named attributes it's called a **natural join** and written as *****.
- Similarly, there are **left, right, and full outer joins**; written as ⟕, ⟖, and ⟗ respectively.

Join Operations

Join

Inner Join

- Theta Join (⋈) {< , > , = , ≠ , ≤ , ≥}.
- Equi Join (⋈ with =)
- **Natural Join**(*) (common attribute names and domain should exist between two tables)

Outer Join

- Left Outer Join ⟕
- Right Outer Join ⟖
- Full Outer Join ⟗

Equi Join

table1

num	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

num	M	ID	N
2	b	2	p

```
Select *
From table1, table 2
Where num=id;
```

What is the equivalent relational algebra expression ?

Equi Join

table1

num	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

Select *
From table1, table 2
Where num=id;

num	M	ID	N
2	b	2	p

result <- table1 $\bowtie_{num=id}$ table2

result

num	M	ID	N
2	b	2	p

Question

table1

num	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

result

num	M	ID	N
2	b	2	p

result <- table1 $\bowtie_{num=id}$ table2

temp <- table1 x table2
result <- $\sigma_{num=ID}$ (temp)

Are these logically equivalent?

Natural Join

table1

ID	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

result <- table1 * table2

result

ID	M	N
2	b	p

Question

table1

ID	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

result

ID	M	N
2	b	p

result <- table1 * table2

temp1 <- table1 x table2
temp2 <- $\sigma_{\text{table1.ID=table2.ID}}$ (temp1)
result <- $\pi_{\text{(table1.ID,M,N)}}$ (temp2)

Are these logically equivalent?

What will be the output of following Relational Algebra Expression

$$\text{Courses} \bowtie \text{HoD}$$

Courses		
CID	**Course**	**Dept**
CS01	Database	CS
ME01	Mechanics	ME
EE01	Electronics	EE

HoD	
Dept	**Head**
CS	Alex
ME	Maya
EE	Mira

What will be the output of following Relational Algebra Expression

$$\text{Courses} \bowtie \text{HoD}$$

Courses		
CID	**Course**	**Dept**
CS01	Database	CS
ME01	Mechanics	ME
EE01	Electronics	EE

HoD	
Dept	**Head**
CS	Alex
ME	Maya
EE	Mira

Courses ⋈ HoD			
Dept	**CID**	**Course**	**Head**
CS	CS01	Database	Alex
ME	ME01	Mechanics	Maya
EE	EE01	Electronics	Mira

Natural Join

table1

ID	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

result <- table1 $\bowtie$ table2

OR

result <- table1 $*$ table2

result

ID	M	N
2	b	p

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.

Problem to solve

Consider three tables

- SAILORS(Sal_ID, SalName, Rating, Age)
- RESERVES(Sal_ID , Boat-ID, Rdate)
- BOATS(Boat-ID, BoatName, Color)

Write Relational Algebra express for the following

i. Find all the names of Sailors who have reserved boat with **ID 2**
ii. Find names of sialors who have reserved RED boat
iii. Find the colors of the boat reserved by **Avinash**

Problem to solve

```
SQL> select * from SAILORS;

    SAL_ID SALNAME                  RATING        AGE
---------- -------------------- ---------- ----------
       101 Avinash                     200         19
       102 Balaji                      150         18
       103 Dinesh                      150         18
```

```
SQL> select * from BOATS;

   BOAT_ID BOATNAME             COLOR
---------- -------------------- ------
         1 Kaveri               Blue
         2 Ganga                Red
```

```
SQL> select * from RESERVES;

    SAL_ID    BOAT_ID RDATE
---------- ---------- ----------
       101          1 12-1-2016
       102          2 18-2-2016
       103          2 25-2-2016
```

Problem to solve

Consider three tables
SAILORS(Sal_ID, SalName, Rating, Age)
RESERVES(Sal_ID , Boat-ID, Rdate)
BOATS(Boat-ID, BoatName, Color)

Write Relational Algebra express for the following

i. Find all the names of Sailors who have reserved boat with **ID 2**

```
SQL> select * from RESERVES;

    SAL_ID    BOAT_ID RDATE
---------- ---------- ----------
       101          1 12-1-2016
       102          2 18-2-2016
       103          2 25-2-2016
```

```
SQL> select * from SAILORS;

    SAL_ID SALNAME      RATING        AGE
---------- ---------- ---------- ----------
       101 Avinash       200         19
       102 Balaji        150         18
       103 Dinesh        150         18
```

OUTPUT

Balaji
Dinesh

Problem to solve

Consider three tables
SAILORS(Sal_ID, SalName, Rating, Age)
RESERVES(Sal_ID , Boat-ID, Rdate)
BOATS(Boat-ID, BoatName, Color)
Write Relational Algebra express for the following

i. Find all the names of Sailors who have reserved boat with **ID 2**

```
SQL> select * from RESERVES;

    SAL_ID    BOAT_ID RDATE
---------- ---------- ----------
       101          1 12-1-2016
       102          2 18-2-2016
       103          2 25-2-2016
```

temp1 <- $\sigma_{Boat\_ID=2}$ (RESERVES)

```
SELECT *
FROM RESERVES
WHERE Boat_ID = 2;
```

temp1

SAL_ID	BOAT_ID	RDATE
102	2	18-2-2016
103	2	25-2-2016

Problem to solve

Consider three tables

SAILORS(Sal_ID, SalName, Rating, Age)

RESERVES(Sal_ID , Boat-ID, Rdate)

BOATS(Boat-ID, BoatName, Color)

Write Relational Algebra express for the following

i. Find all the names of Sailors who have reserved boat with **ID 2**

```
SQL> select * from RESERVES;

    SAL_ID    BOAT_ID RDATE
---------- ---------- ----------
       101          1 12-1-2016
       102          2 18-2-2016
       103          2 25-2-2016
```

temp1 <- $\sigma_{Boat\_ID=2}$(RESERVES)

temp2 <- π_{SAL_ID}(temp1)

```
SELECT SAIL_ID
FROM RESERVES
WHERE Boat_ID = 2;
```

temp2

SAL_ID
102
103

Problem to solve

i. Find all the names of Sailors who have reserved boat with **ID 2**

```
SQL> select * from RESERVES;

    SAL_ID    BOAT_ID RDATE
---------- ---------- ----------
       101          1 12-1-2016
       102          2 18-2-2016
       103          2 25-2-2016
```

```
SQL> select * from SAILORS;

    SAL_ID SALNAME        RATING        AGE
---------- ---------- ---------- ----------
       101 Avinash           200         19
       102 Balaji            150         18
       103 Dinesh            150         18
```

temp1 <- $\sigma_{Boat\_ID=2}$ (RESERVES)

temp2 <- π_{SAL_ID}(temp1)

temp3 <- SAILORS $\bowtie_{SAILORS.Sal\_ID=temp2.Sal\_ID}$ (temp2)

temp3

SAL_ID	SALNAME	RATING	AGE
102	Balaji	150	18
103	Dinesh	150	18

```
SELECT *
FROM SAILORS
NATURAL JOIN
(SELECT SAL_ID
FROM RESERVES
WHERE Boat_ID = 2);
```

Problem to solve

i. Find all the names of Sailors who have reserved boat with **ID 2**

```
SQL> select * from RESERVES;

    SAL_ID     BOAT_ID RDATE
---------- ---------- ----------
       101          1 12-1-2016
       102          2 18-2-2016
       103          2 25-2-2016
```

```
SQL> select * from SAILORS;

    SAL_ID SALNAME         RATING        AGE
---------- --------------- ---------- ----------
       101 Avinash                200         19
       102 Balaji                 150         18
       103 Dinesh                 150         18
```

temp1 <- $\sigma_{Boat\_ID=2}$ (RESERVES)

temp2 <- π_{SAL_ID}(temp1)

temp3 <- SAILORS $\bowtie$ SAILORS.Sal_ID=temp2.Sal_ID (temp2)

result <- π_{SAL_Name} (temp3)

result

SALNAME
Balaji
Dinesh

SELECT SalName
FROM SAILORS
NATURAL JOIN
(SELECT SAL_ID
FROM RESERVES
WHERE Boat_ID = 2);

Problem to solve

i. Find all the names of Sailors who have reserved boat with **ID 2**

```
SQL> select * from RESERVES;

    SAL_ID    BOATID RDATE
---------- ---------- ----------
       101          1 12-1-2016
       102          2 18-2-2016
       103          2 25-2-2016
```

```
SQL> select * from SAILORS;

    SAL_ID SALNAME    RATING           AGE
---------- ---------- ---------- ----------
       101 Avinash           200         19
       102 Balaji            150         18
       103 Dinesh            150         18
```

temp1 <- $\sigma_{Boat\_ID=2}$ (RESERVES)

temp2 <- π_{SAL_ID}(temp1)

temp3 <- SAILORS $\bowtie_{SAILORS.Sal\_ID=temp2.Sal\_ID}$ (temp2)

result <- π_{SAL_Name} (temp3)

```
SQL> select SalName from SAILORS, RESERVES where BOAT_ID=2 and SAILORS.Sal_ID=RE
SERVES.Sal_ID;

SALNAME
--------------------
Balaji
Dinesh
```

SAILORS(Sal_ID, SalName, Rating, Age), RESERVES(Sal_ID , Boat-ID, Rdate), BOATS(Boat-ID, BoatName, Color)

Write Relational Algebra express for the following

ii. Find names of sailors who have reserved RED boat

```
SQL> select * from BOATS;

   BOAT_ID BOATNAME                  COLOR
---------- ------------------------- ------
         1 Kaveri                    Blue
         2 Ganga                     Red
```

```
SQL> select * from RESERVES;

    SAL_ID    BOAT_ID RDATE
---------- ---------- ----------
       101          1 12-1-2016
       102          2 18-2-2016
       103          2 25-2-2016
```

```
SQL> select * from SAILORS;

    SAL_ID SALNAME    RATING        AGE
---------- ---------- ------ ----------
       101 Avinash       200         19
       102 Balaji        150         18
       103 Dinesh        150         18
```

OUTPUT

Balaji
Dinesh

SAILORS(Sal_ID, SalName, Rating, Age), RESERVES(Sal_ID, Boat-ID, Rdate), BOATS(Boat-ID, BoatName, Color)

Write Relational Algebra express for the following

ii. Find the colors of the boat reserved by **Avinash**

```
SQL> select * from SAILORS;

    SAL_ID SALNAME              RATING        AGE
---------- --------------- ---------- ----------
       101 Avinash                200         19
       102 Balaji                 150         18
       103 Dinesh                 150         18
```

```
SQL> select * from RESERVES;

    SAL_ID    BOAT_ID RDATE
---------- ---------- ----------
       101          1 12-1-2016
       102          2 18-2-2016
       103          2 25-2-2016
```

```
SQL> select * from BOATS;

   BOAT_ID BOATNAME             COLOR
---------- -------------------- ------
         1 Kaveri               Blue
         2 Ganga                Red
```

```
OUTPUT
--------
Red
```

Join Operations

Join

- Inner Join
  - Theta Join (⋈) {< , > , = , ≠ , ≤ , ≥}.
  - Equi Join (⋈ with =)
  - **Natural Join**(*) (common attribute names and domain should exist between two tables)
- Outer Join
  - Left Outer Join ⟕
  - Right Outer Join ⟖
  - Full Outer Join ⟗

Left Outer Join ⟕

result <- table1 ⟕ table2

table1

ID	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

result

ID	M	ID	N
2	b	2	P
1	a		
4	c		

```
SQL> select *
     from table1 left outer join table2
     on table1.id=table2.id;
```

Right Outer Join ⟖

result <- table1 ⟖ table2

table1

ID	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

result

ID	M	ID	N
2	b	2	P
		3	q
		5	r

```
SQL> select *
     from table1 right outer join table2
     on table1.id=table2.id;
```

Full Outer Join ⟗

result <- table1 ⟗ table2

table1

ID	M
1	a
2	b
4	c

table2

ID	N
2	p
3	q
5	r

result

ID	M	ID	N
2	b	2	p
		3	q
		5	r
1	a		
4	c		

```
SQL> select *
     from table1 full join table2
     on table1.id = table2.id;
```

Relational Algebra Set Operations - semantics

Consider two relations R and S.

- **UNION** of R and S
  the union of two relations is a relation that includes all the tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.
- **INTERSECTION** of R and S
  the intersection of R and S is a relation that includes all tuples that are both in R and S.
- **DIFFERENCE** of R and S
  the difference of R and S is the relation that contains all the tuples that are in R but that are not in S.

For set operations to function correctly the relations R and S must be union compatible. Two relations are union compatible if

- they have the same number of attributes
- the domain of each attribute in column order is the same in both R and S.

Set Operation – Union U

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

result <- R U S

result

A	1
B	2
C	2
D	3
E	5
F	4
E	4

Set Operation – Intersection ∩

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

result <- R ∩ S

result

A	1
D	3

Set Operation – Difference –

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

result <- R – S

B	2
F	4
E	5

result <- S – R

C	2
E	4

Problem to Solve: Writing Relational Algebra Expression

Consider the following three tables

- STUDENT(StudNum, StudName)
- PROJECT(ProjNum, ProjArea)
- ASSIGED_TO(StudNum,ProjNum)

i. Obtain student number and student name of all students who are working on both the projects having project number 75 and 81

ii. Obtain student number and student name of all those students who do not work on project number 68

iii. Obtain the student number and student name of all those students who are working on project with name "Database"

iv. Obtain student number and student name of all students other than the student with number 554 who works on atleast one project.

Problem to Solve: Writing Relational Algebra Expression

Consider the following three tables

- STUDENT(StudNum, StudName), PROJECT(ProjNum, ProjArea), ASSIGED_TO(StudNum,ProjNum)

i. Obtain student number and student name of all students who are working on both the projects having project number 75 and 81

STUDENT

StudNum	StudName
554	Avinash
555	Balaji
556	Chandan
557	Dinesh
558	Harish

PROJECT

ProjNum	ProjArea
56	Java
68	Database
75	Database
81	Database

ASSIGNED_TO

Stud Num	Proj Num
554	56
555	68
556	75
556	81
557	75

Problem to Solve: Writing Relational Algebra Expression

i. Obtain student number and student name of all students who are working on **both** the projects having **project number 75 and 81**

STUDENT

StudNum	StudName
554	Avinash
555	Balaji
556	Chandan
557	Dinesh
558	Harish

ASSIGNED_TO

Stud Num	Proj Num
554	56
555	68
556	75
556	81
557	75

result

----

556 Chandan

Problem to Solve: Writing Relational Algebra Expression

i. Obtain student number and student name of all students who are working on **both** the projects having **project number 75 and 81**

STUDENT

StudNum	StudName
554	Avinash
555	Balaji
556	Chandan
557	Dinesh
558	Harish

ASSIGNED_TO

Stud Num	Proj Num
554	56
555	68
556	75
556	81
557	75

result

----

556 Chandan

temp1 <- $\sigma_{ProjNum=\mathbf{75}}$(ASSIGNED_TO)

temp2 <- $\pi_{StudNum}$(temp1)

temp3 <- $\sigma_{ProjNum=\mathbf{81}}$(ASSIGNED_TO)

temp4 <- $\pi_{StudNum}$(temp3)

temp5 <- temp2 $\cap$ temp2

result <- STUDENT $\bowtie_{STUDENT.StudNum=temp5.StudNum}$ (temp5)

Problem to Solve: Writing Relational Algebra Expression

ii. Obtain student number and student name of all those students who do not work on project number 68

STUDENT

StudNum	StudName
554	Avinash
555	Balaji
556	Chandan
557	Dinesh
558	Harish

ASSIGNED_TO

Stud Num	Proj Num
554	56
555	68
556	75
556	81
557	75

result

554	Avinash
556	Chandan
557	Dinesh
558	Harish

Problem to Solve: Writing Relational Algebra Expression

i. Obtain student number and student name of all those students who do not work on project number 68

STUDENT

StudNum	StudName
554	Avinash
555	Balaji
556	Chandan
557	Dinesh
558	Harish

ASSIGNED_TO

Stud Num	Proj Num
554	56
555	68
556	75
556	81
557	75

result

554	Avinash
556	Chandan
557	Dinesh
558	Harish

temp1 <- $\sigma_{ProjNum=\mathbf{68}}$(ASSIGNED_TO)
temp2 <- $\pi_{StudNum}$(temp1)
temp3 <- $\pi_{StudNum}$(STUDENT)
temp4 <- temp3 – temp2

result <- STUDENT $\bowtie_{STUDENT.StudNum=temp4.StudNum}$ (temp4)

Homework Problem : Writing Relational Algebra Expression

iii. Obtain the student number and student name of all those students who are working on project with name "Database"

STUDENT

StudNum	StudName
554	Avinash
555	Balaji
556	Chandan
557	Dinesh
558	Harish

PROJECT

ProjNum	ProjArea
56	Java
68	Database
75	Database
81	Database

ASSIGNED_TO

Stud Num	Proj Num
554	56
555	68
556	75
556	81
557	75

result

555	Balaji
556	Chandan
557	Dinesh

Homework Problem: Writing Relational Algebra Expression

iv. Obtain student number and student name of all students other than the student with number 554 who works on atleast one project.

STUDENT

StudNum	StudName
554	Avinash
555	Balaji
556	Chandan
557	Dinesh
558	Harish

PROJECT

ProjNum	ProjArea
56	Java
68	Database
75	Database
81	Database

ASSIGNED_TO

Stud Num	Proj Num
554	56
555	68
556	75
556	81
557	75

result

555	Balaji
556	Chandan
557	Dinesh

Relational Algebra : Division Operation ÷

The division operator is used for queries which involve the ‘all’ qualifier such as

- “Which persons have a bank account at ALL the banks in the c ountry?”
- “Which students are registered on ALL the courses given by So nthos?”
- “Which students are registered on ALL the courses that are tau ght in period 1?”
- Find sailors who have reserved ALL boats

R ÷ S is used when we wish to express queries with “ALL”

Relational Algebra : Division Operation ÷

The division operator takes as input two relations, called the dividend relation (*a* on scheme A) and the divisor relation (b on scheme *B*) such that all the attributes in *B* also appear in *A* and *B* is not empty. The output of the division operation is a relation on scheme *A* with all the attributes common with B.

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B

pno
p2
p4

A ÷ B

sno
s1
s4

Relational Algebra : Division Operation ÷

The division operator takes as input two relations, called the dividend relation (*a* on scheme A) and the divisor relation (b on scheme *B*) such that all the attributes in *B* also appear in *A* and *B* is not empty. The output of the division operation is a relation on scheme *A* with all the attributes common with B.

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B

pno
p2
p4

A ÷ B

sno
s1
s4

Relational Algebra : Division Operation ÷

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B

pno
p1
p2
p4

$\boxed{A \div B}$

sno
s1

Relational Algebra : Division Operation ÷

Completed

Student	Task
Feroz	Database1
Feroz	Database2
Feroz	Compiler1
Eshwar	Database1
Sara	Database1
Sara	Database2
Eshwar	Compiler1

DBProject

Task
Database1
Database2

What is the Output of the following relational Algebra Expression

Completed ÷ DBProject

Relational Algebra : Division Operation ÷

Completed

Student	Task
Feroz	Database1
Feroz	Database2
Feroz	Compiler1
Eshwar	Database1
Sara	Database1
Sara	Database2
Eshwar	Compiler1

DBProject

Task
Database1
Database2

Completed ÷ DBProject

Student
Feroz
Sara

Relational Algebra : Division Operation ÷

Completed

Student	Task
F	D1
F	D2
F	C1
E	D1
E	C1

DBProject

Task
D1
D2

Completed ÷ DBProject

Student
F

Is the following two relational algebra expressions logically equivalent ?

$T \leftarrow \text{Completed} \div \text{DBProject}$

$T_1 \leftarrow \pi_{\text{Student}}(\text{Completed})$
$T_2 \leftarrow T_1 \times \text{DBProject}$
$T_3 \leftarrow T_2 - \text{Completed}$
$T_4 \leftarrow \pi_{\text{Student}}(T_3)$
$T \leftarrow T_1 - T_4$

Write relational Algebra Expresion

Find all bank customers who have account in all Branches of Bommasandra

Account

cid	branch_id	acct_no	balance
1	52	8103	43101.45
3	53	4826	752.80
1	**53**	7898	48206.10
2	59	2135	468923.06
1	**59**	1290	456.50
2	54	0073	1006.28

Branch

branch_id	branch_city
51	Belgaum
52	Bijapur
53	**Bommasandra**
54	Hubli
55	Bijapur
59	**Bommasandra**

Customer

cid	c_name
1	**Harish**
2	Triveni
3	Eshwar

RESULT

c_name
Harish

Find all bank customers who have account in all Branches in Bommasandra

Account

cid	branch_id	acct_no	balance
1	52	8103	43101.45
3	53	4826	752.80
1	**53**	7898	48206.10
2	59	2135	468923.06
1	**59**	1290	456.50
2	54	0073	1006.28

Branch

branch_id	branch_city
51	Belgaum
52	Bijapur
53	**Bommasandra**
54	Hubli
55	Bijapur
59	**Bommasandra**

Customer

cid	c_name
1	**Harish**
2	Triveni
3	Eshwar

BommB $\leftarrow \pi_{\text{branch\_id}}(\sigma_{\text{branch\_city}=\text{'Bommasandra'}}(\text{Branch}))$ *--find all branches located in Bommansandra*

BommB

branch_id
53
59

Find all bank customers who have account in all Branches in Bommasandra

Account

cid	branch_id	acct_no	balance
1	52	8103	43101.45
3	53	4826	752.80
1	**53**	7898	48206.10
2	59	2135	468923.06
1	**59**	1290	456.50
2	54	0073	1006.28

Branch

branch_id	branch_city
51	Belgaum
52	Bijapur
53	**Bommasandra**
54	Hubli
55	Bijapur
59	**Bommasandra**

Customer

cid	c_name
1	**Harish**
2	Triveni
3	Eshwar

BommB ← $\pi_{\text{branch\_id}}(\sigma_{\text{branch\_city='Bommasandra'}}(\text{Branch}))$ *--find all branches located in Bommansandra*

CB ← $\pi_{\text{c\_name,branch\_id}}(\text{Customer} * \text{Account})$ *-- find all customers' branches*

BommB

branch_id
53
59

CB

c_name	branch_id
Harish	52
Eshwar	53
Harish	53
Triveni	59
Harish	59
Triveni	54

Find all bank customers who have account in all Branches in Bommasandra

Account

cid	branch_id	acct_no	balance
1	52	8103	43101.45
3	53	4826	752.80
1	**53**	7898	48206.10
2	59	2135	468923.06
1	**59**	1290	456.50
2	54	0073	1006.28

Branch

branch_id	branch_city
51	Belgaum
52	Bijapur
53	**Bommasandra**
54	Hubli
55	Bijapur
59	**Bommasandra**

Customer

cid	c_name
1	**Harish**
2	Triveni
3	Eshwar

BommB $\leftarrow \pi_{\text{branch\_id}}(\sigma_{\text{branch\_city='Bommasandra'}}(\text{Branch}))$ *--find all branches located in Bommansandra*

CB $\leftarrow \pi_{\text{c\_name,branch\_id}}(\text{Customer} * \text{Account})$ *-- find all customers' branches*

RESULT $\leftarrow$ CB $\div$ BinB *-- divide to get those customers with an account in every Bommasandra branch*

BommB

branch_id
53
59

CB

c_name	branch_id
Harish	52
Eshwar	53
Harish	53
Triveni	59
Harish	59
Triveni	54

RESULT

c_name
Harish

Table 6.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{<\text{selection condition}>}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{<\text{attribute list}>}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{<\text{join condition}>} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{<\text{join condition}>} R_2$, OR $R_1 \bowtie_{(<\text{join attributes 1}>),\ (<\text{join attributes 2}>)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{<\text{join condition}>} R_2$, OR $R_1 *_{(<\text{join attributes 1}>),\ (<\text{join attributes 2}>)} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Query Tree Notation

- Query Tree
  - An internal data structure to represent a query
  - Standard technique for estimating the work involved in executing the query, the generation of intermediate results, and the optimization of execution
  - Nodes stand for operations like selection, projection, join, renaming, division, ….
  - Leaf nodes represent base relations
  - A **tree** gives a **good visual feel of the complexity of the query** and the operations involved
  - Algebraic Query Optimization consists of rewriting the query or modifying the query tree into an equivalent tree.

Query Tree Notation: Example

- Write query: For every project located in 'Surat', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

PROJECT

Pname	Pnumber	Plocation	Dnum

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Salary	Super_ssn	Dno

Query Tree Notation: Example

- Example: For every project located in 'Surat', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

PROJECT

Pname	Pnumber	Plocation	Dnum

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

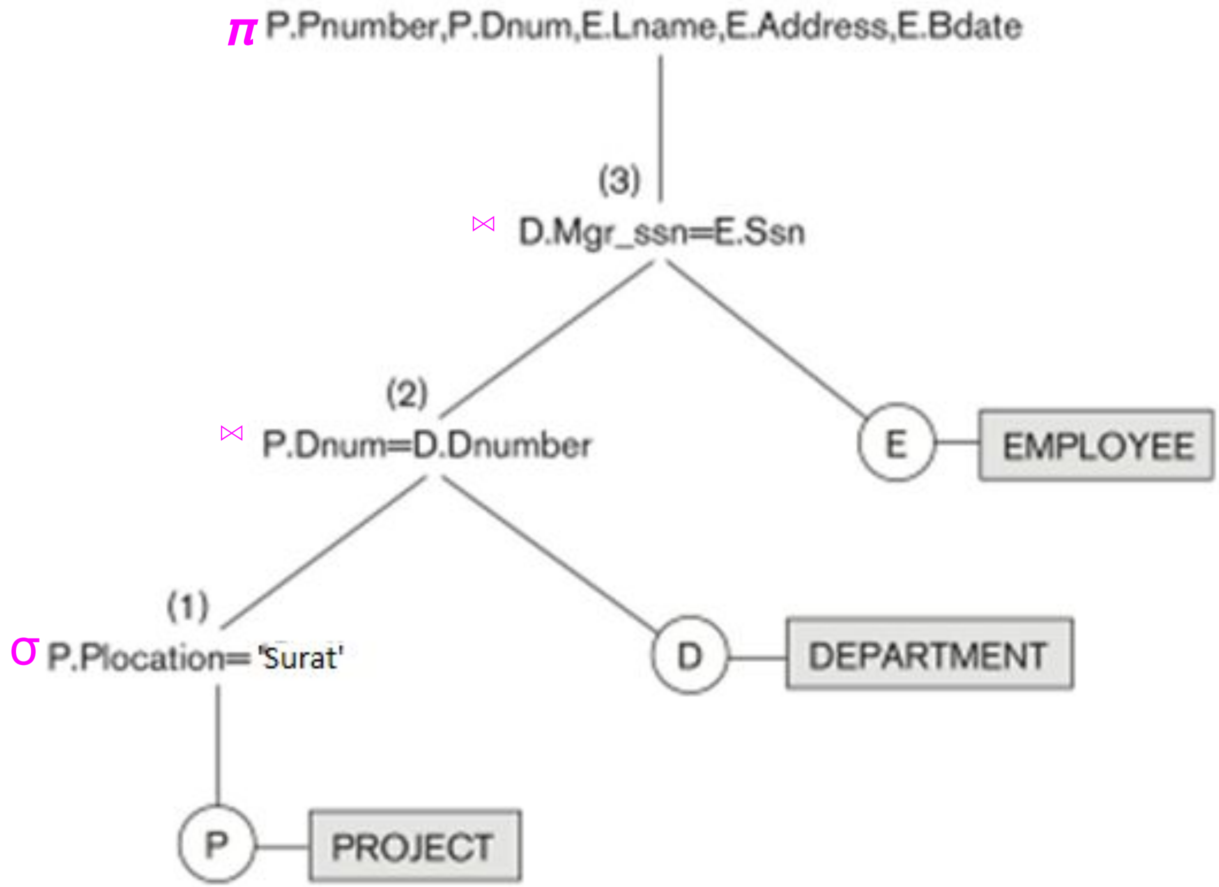
EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Salary	Super_ssn	Dno

$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}\left(\left(\sigma_{\text{Plocation='Surat'}}(\text{Project}) \bowtie_{\text{Dnum=Dnumber}} \text{Department}\right) \bowtie_{\text{Mgr\_ssn=Ssn}} \text{Employee}\right)$$

Query Tree Notation: Example

$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}$ ($(\sigma_{\text{Plocation='Surat'}}(\text{Project}) \bowtie_{\text{Dnum=Dnumber}} \text{Department}) \bowtie_{\text{Mgr\_ssn=Ssn}} \text{Employee}$)



Generalized Projection π

- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\pi_{F1, F2, \ldots, Fn}(E)$$

- E is any relational-algebra expression
- Each of F_1, F_2, …, F_n are are arithmetic expressions involving constants and attributes in the schema of E.

Generalized Projection π

- Given relation

credit-info(CustomerName, Limit, CreditBalance)

Customer-name	**Limit**	**CreditBalance**
Avinash	2000	500
Balaji	700	100
Chandan	1500	1000

Find how much money each person can spend:

Generalized Projection π

- Given relation

credit-info(CustomerName, Limit, CreditBalance)

Customer-name	Limit	CreditBalance
Avinash	2000	500
Balaji	700	100
Chandan	1500	1000

Find how much money each person can spend:

result <- $\Pi_{\text{customer-name, } (\textbf{limit – credit-balance})}$ *(credit-info)*

Customer-name	Limit - CreditBalance
Avinash	1500
Balaji	600
Chandan	500

Generalized Projection π

- Another Example

Consider a relation

EMPLOYEE(EMP-ID, Salary, Deduction, Years-of-Service)

A report may be required to show:

- Net_salary = Salary – Deduction
- Bonus = 2000 * Years-of-Service
- Tax = Salary * 25%

Then a generalized projection combined with renaming may be:

report <- $\rho_{\text{(Net\_salary, Bonus, Tax)}}$

$(\pi_{\text{EMP-ID, (Salary – Deduction), (2000 * Years-of-Service), (Salary * 0.25)}}$ (EMPLOYEE))

Aggregate Function $\mathcal{F}$ (script F)

- Aggregate functions return a single value, calculated from values in a column.

Useful aggregate functions:

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

Student Table

usn	name	dep_num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	arvind	20	90

$$\mathcal{F}_{\text{MAX marks}}(\text{Student})$$

MAX_marks
100

Aggregate Function $\mathcal{F}$ (script F)

Student Table

usn	name	dep_ num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	arvind	20	90

$\mathcal{F}_{\text{COUNT usn, AVERAGE marks}}(\text{Student})$

COUNT_usn	AVERAGE_marks
5	75

Using Grouping with Aggregation

<grouping attribute> $\mathcal{F}$ <aggregate function list>

- <grouping attribute> is a list of attributes of the relation specified in R and <aggregate function list> is a list of (<function> <attribute>)

Student Table

usn	name	dep_num	marks
1BM14CS001	Avinash	10	100
1BM14CS002	Balaji	10	80
1BM14CS003	Chandan	10	45
1BM14CS004	Dinesh	10	60
1BM14IS001	arvind	20	90

$${}_{dep\_num}\mathcal{F}_{COUNT\ usn,\ AVERAGE\ marks}(Student)$$

dep-num	COUNT_usn	AVERAGE_marks
10	4	71.25
20	1	90

Problem to Solve: Writing Relational Algebra Expression

Consider the following three tables

- SALESPERSON(SalesPersonID,Name)
- TRIP(SalesPersonID, From, To, Trip-ID)
- EXPENSE(TripID, Amount)

i. Print the total trip expenses incurred by sales person with ID 504

ii. Give the trip details for the trip that exceeded Rs. 10,000/-

iii. Print the sales person ID and Name of the sales men who took trips to Delhi

Problem to Solve: Writing Relational Algebra Expression

Consider the following three tables

- SALESPERSON(SalesPersonID,Name)
- TRIP(SalesPersonID, From, To, Trip-ID)
- EXPENSE(TripID, Amount)

SALESPERSON

SalesPersonID	Name
504	Avinash
505	Balaji
506	Chandan

EXPENSE

Trip-ID	Amount
10	10000
11	8000
12	15000

TRIP

SalesPersonID	From	To	Trip-ID
504	Chennai	Delhi	10
504	Bangalore	Bombay	11
505	Bangalore	Srinagar	12

Problem to Solve: Writing Relational Algebra Expression

i. Print the total trip expenses incurred by sales person with ID 504

EXPENSE

Trip-ID	Amount
10	10000
11	8000
12	15000

TRIP

SalesPersonID	From	To	Trip-ID
504	Chennai	Delhi	10
504	Bangalore	Bombay	11
505	Bangalore	Srinagar	12

result
18000

Problem to Solve: Writing Relational Algebra Expression

i. Print the total trip expenses incurred by sales person with ID 504

EXPENSE

Trip-ID	Amount
10	10000
11	8000
12	15000

TRIP

SalesPersonID	From	To	Trip-ID
504	Chennai	Delhi	10
504	Bangalore	Bombay	11
505	Bangalore	Srinagar	12

result
18000

temp1 <- $\sigma_{\text{SalespersonID}=\mathbf{504}}$(TRIP)

temp2 <- $\pi_{\text{Trip-ID}}$(temp1)

temp3 <- EXPENSE $\bowtie_{\text{EXPENSE.Trip-ID=temp2.Trip-ID}}$ (temp2)

result <- $\mathcal{F}_{\text{SUM Amount}}$ (temp3)

Problem to Solve: Writing Relational Algebra Expression

ii. Give the **trip details** for the trip that exceeded Rs. 10,000/-

EXPENSE

Trip-ID	Amount
10	10000
11	8000
12	15000

TRIP

SalesPersonID	From	To	Trip-ID
504	Chennai	Delhi	10
504	Bangalore	Bombay	11
505	Bangalore	Srinagar	12

result

504	Chennai	Delhi	10	10000
505	Bangalore	Srinagar	12	15000

Problem to Solve: Writing Relational Algebra Expression

ii. Give the **trip details** for the trip that exceeded Rs. 10,000/-

EXPENSE

Trip-ID	Amount
10	10000
11	8000
12	15000

TRIP

SalesPersonID	From	To	Trip-ID
504	Chennai	Delhi	10
504	Bangalore	Bombay	11
505	Bangalore	Srinagar	12

result

504	Chennai	Delhi	10	10000
505	Bangalore	Srinagar	12	15000

temp1 <- $\sigma_{\text{Amount} > \mathbf{10000}}$(EXPENSE)

result <- $\pi_{\text{(TRIP.SalesPersonID, TRIP.From, TRIP.To, TRIP.Trip-ID,temp2.Amount)}}$ (TRIP $\bowtie_{\text{TRIP.Trip-ID=temp1.Trip-ID}}$ (temp1))

Problem to Solve: Writing Relational Algebra Expression

iii. Print the sales person ID and Name of the sales men who took trips to Delhi

SALESPERSON

SalesPersonID	Name
504	Avinash
505	Balaji
506	Chandan

TRIP

SalesPersonID	From	To	Trip-ID
504	Chennai	Delhi	10
504	Bangalore	Bombay	11
505	Bangalore	Srinagar	12

result
----
504 Avinash

Problem to Solve: Writing Relational Algebra Expression

iii. Print the sales person ID and Name of the sales men who took trips to Delhi

SALESPERSON

SalesPersonID	Name
504	Avinash
505	Balaji
506	Chandan

TRIP

SalesPersonID	From	To	Trip-ID
504	Chennai	Delhi	10
504	Bangalore	Bombay	11
505	Bangalore	Srinagar	12

result
----
504 Avinash

temp1 <- $\sigma_{To=\text{`}\mathbf{Delhi}\text{'}}$(TRIP)

temp2 <- $\pi_{\text{SalesPerson-ID}}$(temp1)

temp3<-SALESPERSON $\bowtie_{\text{SALESPERSON.SalesPersonID=temp2.SalesPerson-ID}}$ (temp2)

result <- $\pi_{\text{SalesPersonID, Name}}$ (temp3)

Problem to Solve: Writing Relational Algebra Expression

Consider the following relational schema describing a movie database

- Schedule(Theater, Title, Time)
- Movies(Title, Director, Actor)
- Produced(Producer, Title)
- See(Spectator, Title)
- Liked(Spectator, Title)
- A movie is directed by only one director but can be produced by several Producers. A spectator may like a movie without having seen it. Write the following two queries in both Relational Algebra and SQL.

Note: In relational algebra you must use the expression form and are not allowed to use linear sequence or expression trees. You are also NOT allowed renaming of relations. You may use renaming of attributes. You may use numerical comparisons (e.g. R:A > 5) in both SQL and Relational Algebra.

- List the people who liked movies that they have not seen
- List the producers who produced a movie that does not appear in a theater.

Problem to Solve: Writing Relational Algebra Expression

See(Spectator, Title)
Liked(Spectator, Title)

- List the people who liked movies that they have not seen

$$\Pi_{spectator}(Liked - See)$$

```
SELECT Spectator
FROM (SELECT Spectator, Title
        FROM Liked
            EXCEPT
        SELECT Spectator, Title
        FROM See
       )
```

Problem to Solve: Writing Relational Algebra Expression

Schedule(Theater, Title, Time)
Movies(Title, Director, Actor)
Produced(Producer, Title)

- List the producers who produced a movie that does not appear in a theater.

$$\Pi_{Producer}(Produced \bowtie (\Pi_{title}(Movies) - \Pi_{title}(Schedule))$$

```
SELECT Producer
FROM Produced
WHERE title IN (SELECT title
                FROM MOVIES
                     EXCEPT
                SELECT title
                FROM Schedule
               )
```

Relational Algebra Operations

- Unary Operations - operate on one relation. These include select, project and rename operators.
- Binary Operations - operate on pairs of relations. These include union, set difference, intersection, division, cartesian product, join, equality join, natural join, Left Outer join, Right outer join and full outer join.

Thank You for Your Time and Attention !

Students Should read through the

-Relational algebra example queries given in the ELMARSI and NAVATHE text book in chapter 6 of section 6.5

-Relational Model constraints and Relational database schema, Update Transactions and Dealing with constraint violation from ELMARSI and NAVATHE text book in chapter 5 of sections 5.2 and 5.3