UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FAKULTAS ILMU KOMPUTER

Slide 13

Relational Algebra

CSF2600700 - BASIS DATA

SEMESTER GENAP 2017/2018

References

- Elmasri 6th edition: Chapter 6
- Stanford Database (CS145) Lecture 16

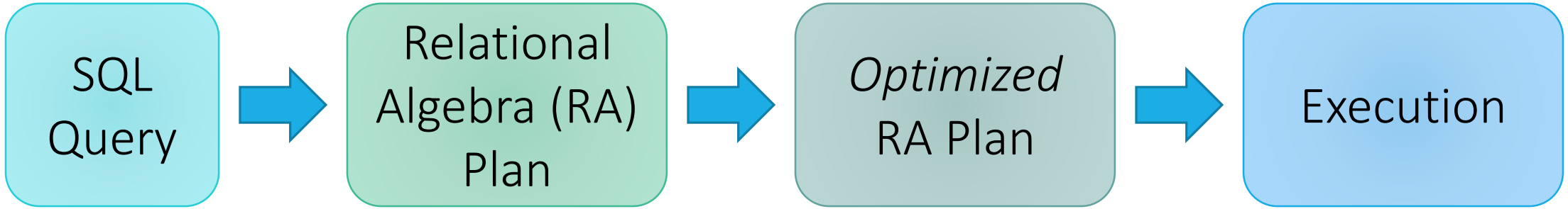
Introduction

- Relational Algebra (RA): The basic set of operations for relational model

- Important for:
  - Provides a formal foundation for relational model operations
  - Used as basis for implementing and optimizing queries by RDBMS
  - Incorporated into the SQL for RDBMS

RDBMS Architecture

How does a SQL engine work ?



Declarative query (from user)

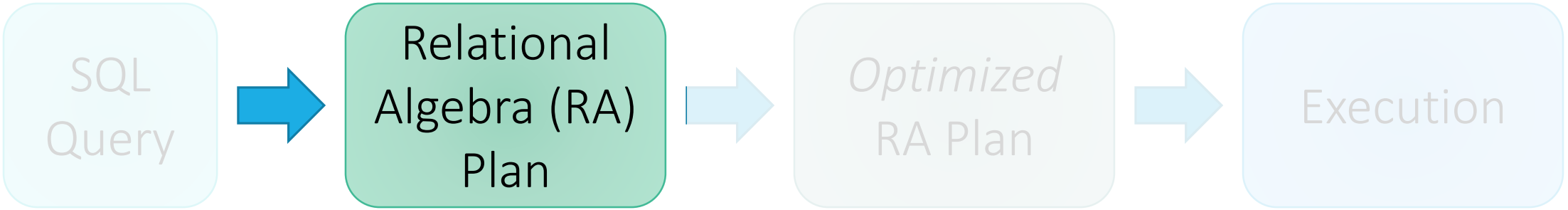
Translate to relational algebra expression

Find logically equivalent- but more efficient- RA expression

Execute each operator of the optimized plan!

RDBMS Architecture

How does a SQL engine work ?



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

Relational Algebra Operations

- Set Operations
  - UNION ($\cup$)
  - INTERSECTION ($\cap$)
  - SET DIFFERENCE ($-$)
  - CARTESIAN/CROSS PRODUCT ($\times$)

  } Binary Operations

- Operations for specific Relational DB
  - SELECT (σ)
  - PROJECT (π)
  - RENAME (ρ)

  } Unary Operations

  - JOIN ($\bowtie$)

- Other
  - DIVISION ($\div$)

  } Binary Operations

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 Operations

SELECT Operations

Students(sid,sname,gpa)

- Returns all tuples which satisfy a condition
- Notation: $\sigma_{<condition>}(R)$
- Examples:
  - $\sigma_{Salary > 40000}$ (Employee)
  - $\sigma_{name = \text{“Smith”}}$ (Employee)
- The $< condition >$ can be:
  - $< attr\ name >< operator >< constant\ value >$
    or
  - $< attr\ name >< operator >< attr\ name >$
- Operator is one of =, <, ≤, >, ≥, <>

SQL:

```
SELECT *
FROM Students
WHERE gpa > 3.5;
```

RA:

$$\sigma_{gpa > 3.5}(Students)$$

SELECT Operations

Another example:

SSN	Name	Salary
1234545	John	200000
5423341	Smith	600000
4352342	Fred	500000

$\sigma_{Salary > 40000}$ (Employee)

SSN	Name	Salary
5423341	Smith	600000
4352342	Fred	500000

PROJECT Operation

Students(sid,sname,gpa)

- Select certain columns and eliminates the other columns, then removes duplicates
- Notation: $\pi_{<attribute\ list>}(R)$
- Example:
  - $\pi_{\text{sname, gpa}}$ (Students)
  - $\pi_{\text{Name, Salary}}$ (Employee)

SQL:

```
SELECT DISTINCT
  sname,
  gpa
FROM Students;
```

RA:

$$\Pi_{sname,gpa}(Students)$$

PROJECT Operation

Another example:

SSN	Name	Salary
1234545	John	200000
5423341	John	600000
4352342	John	200000

$\pi_{Name,Salary}$ (Employee)

Name	Salary
John	200000
John	600000

Note that RA Operators are Compositional!

Students(sid,sname,gpa)

```
SELECT DISTINCT
  sname,
  gpa
FROM Students
WHERE gpa > 3.5;
```

$$\Pi_{sname,gpa}(\sigma_{gpa>3.5}(Students))$$

$$\sigma_{gpa>3.5}(\Pi_{sname,gpa}(Students))$$

Are these logically equivalent?

How do we represent this query in RA?

Logical Equivalence of RA Plans

- Given relations R(A,B) and S(B,C):
  - Here, projection & selection commute:
    - $\sigma_{A=5}(\Pi_A(R)) = \Pi_A(\sigma_{A=5}(R))$
  - What about here?
    - $\sigma_{A=5}(\Pi_B(R)) \stackrel{?}{=} \Pi_B(\sigma_{A=5}(R))$

Generalized PROJECT Operation

- Also allows functions of attributes to be included in the projection list.

As an example, consider the relation

EMPLOYEE (Ssn, Salary, Deduction, Years_service)

A report may be required to show

Net Salary = Salary – Deduction,
Bonus = 2000 * Years_service, and
Tax = 0.25 * Salary.

Then a generalized projection combined with renaming may be used as follows:

$$\text{REPORT} \leftarrow \rho_{(\text{Ssn, Net\_salary, Bonus, Tax})}(\pi_{\text{Ssn, Salary} - \text{Deduction, } 2000 * \text{Years\_service, } 0.25 * \text{Salary}}(\text{EMPLOYEE})).$$

Rename Operation: See next page

RENAME Operation

Students(sid,sname,gpa)

- RENAME: Changes the schema, not the instance
- Notation:
  - $\rho_{S(B_1,\ldots B_n)}(R)$, or $\rho_S(R)$, or $\rho_{(B_1,\ldots B_n)}(R)$
    - S: the new relation name
    - $B_1, \ldots B_n$: the new attribute names
- **Note: this is shorthand for the proper form (since names, not order matters!):**
  - $\rho_{A1 \rightarrow B1,\ldots,An \rightarrow Bn}(R)$

SQL:

```
SELECT
  sid AS studId,
  sname AS name,
  gpa AS gradePtAvg
FROM Students;
```

RA:

$\rho_{studId,name,gradePtAvg}(Students)$

We care about this operator *because* we are working in a *named perspective*

RENAME Operation

Another example:

$$\rho_{studId,name,gradePtAvg}(Students)$$

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

Students

studId	name	gradePtAvg
001	John	3.4
002	Bob	1.3

- $$\rho_{Stud-Rel(A,B,C)}(Students)$$

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

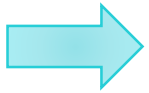
Stud-Rel

A	B	C
001	John	3.4
002	Bob	1.3

(Back to:) Note that RA Operators are Compositional!

Students(sid,sname,gpa)

```
SELECT DISTINCT
  sname,
  gpa
FROM Students
WHERE gpa > 3.5;
```



$$\Pi_{sname,gpa}(\sigma_{gpa>3.5}(Students))$$

$$TEMP \leftarrow \sigma_{gpa>3.5}(Students)$$
$$R(Name, GPA) \leftarrow \pi_{sname,gpa}(TEMP)$$

$$\sigma_{gpa>3.5}(\Pi_{sname,gpa}(Students))$$

$$TEMP \leftarrow \pi_{sname,gpa}(Students)$$
$$R(Name, GPA) \leftarrow \sigma_{gpa>3.5}(TEMP)$$

We can also break down a sequences/compositional RA by specifying intermediate result relations and renaming the relation and its attributes.

Binary Operations

- Two relations $R_1(A_1, A_2, \ldots A_n)$ and $R_2(B_1, B_2, \ldots B_n)$ are said to be **union compatible** if they have the same degree n and if $dom(A_i) = dom(B_i)$ for $1 \leq i \leq n$

- Need union compatibility:
  - UNION, INTERSECTION, SET DIFFERENCE

- Do not need union compatibility:
  - CARTESIAN PRODUCT, JOIN

Example:

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

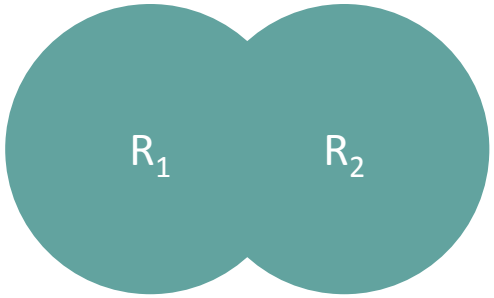
INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

Both have two attributes with the same domain

UNION Operation

- Includes all tuples that are either in R_1 or R_2 or both without duplicate tuples.
- Notation: $R_1 \cup R_2$



- Commutative, Associative

Example: Student ∪ Instructor

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

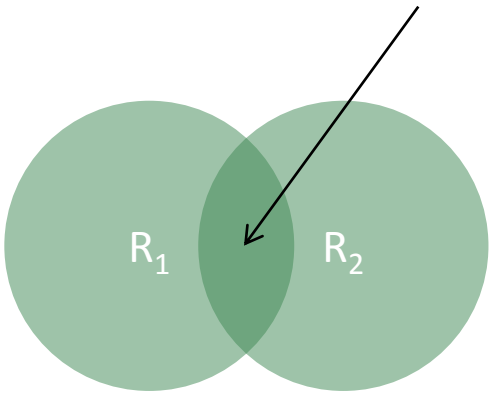
INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

INTERSECTION Operation

- Includes all tuples that are in both R_1 and R_2
- Notation: $R_1 \cap R_2$



- $R_1 \cap R_2 = R_1 - (R_1 - R_2)$
- Commutative, Associative

Example: Student ∩ Instructor

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

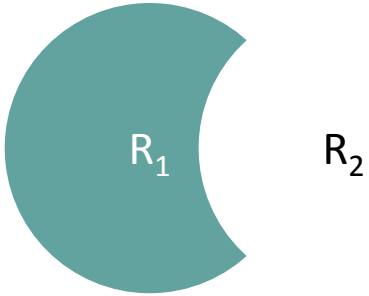
INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

Fn	Ln
Susan	Yao
Ramesh	Shah

DIFFERENCE/MINUS/EXCEPT Operation

- Includes all tuples that are in R but not in S
- Notation: $R - S$



$$R - S =$$
$$\big((R \cup S) - (R - S)\big) - (S - R)$$

Example:

STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

Student – Instructor

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

Instructor – Student

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

CARTESIAN/CROSS PRODUCT

- Combining each tuple in R_1 with each tuple in R_2
- Notation: $R_1 \times R_2$
- Example:
  - Employee × Dependents
  - People × Students
- Rare in practice; mainly used to express joins
- Useful when followed by a selection condition (see example in the next page)

People(ssn,pname,address)
Students(sid,sname,gpa)

SQL:

```
SELECT *
FROM People, Students
```

RA: $People \times Students$

Another example:

```
SELECT *
FROM People, Students
WHERE and pname = sname;
```

People

ssn	pname	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

×

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

$PeopStud \leftarrow People \times Students$

$Result \leftarrow \sigma_{pname=sname}(PeopStud)$

PeopStud

ssn	pname	address	sid	sname	gpa
1234545	John	216 Rosse	001	John	3.4
5423341	Bob	217 Rosse	001	John	3.4
1234545	John	216 Rosse	002	Bob	1.3
5423341	Bob	216 Rosse	002	Bob	1.3

Also can be represented as this single line RA:

$Result \leftarrow \sigma_{pname=sname}(People \times Students)$

Result

ssn	pname	address	sid	sname	gpa
1234545	John	216 Rosse	001	John	3.4
5423341	Bob	216 Rosse	002	Bob	1.3

(Theta) JOIN

- Notation: $R_1 \bowtie_{<join\ condition>} R_2$
  - $< join\ condition > = < condition > AND < condition > \cdots < condition >$
  - Each $< condition >$ in the form $A_i \theta B_j$ and θ is one of $=, <, \leq, >, \geq, <>$
  - A_i is attribute in R_1, B_j is attribute in R_2

- Previous CROSS PRODUCT's example can be replaced by this JOIN operation:
  - $Result \leftarrow People \bowtie_{pname=sname} (Students)$

- This JOIN example is called EQUIJOIN – a JOIN operation where the only operator used is "="

(Natural) JOIN

- A theta JOIN + PROJECT operation.
  - Join two relations, then use projection to select desired columns
- An EQUIJOIN with the removal of superfluous attribute.
  - See example in the next page.

- Notation: $R_1 * R_2$ (or $R_1 \bowtie R_2$ without join condition)

- The attributes to be joined must have the same name in both relations, otherwise we have to rename one of them first.

People

ssn	pname	address
1234545	John	216 Rosse
5423341	Bob	217 Rosse

Students

sid	sname	gpa
001	John	3.4
002	Bob	1.3

(Natural) JOIN

- Example:
  - EQUIJOIN: $People \bowtie_{pname=sname} Students$

ssn	pname	address	sid	sname	gpa
1234545	John	216 Rosse	001	John	3.4
5423341	Bob	216 Rosse	002	Bob	1.3

Superfluous Attribute

  - Natural Join: $People * \rho_{(sid,pname,gpa)} Students$

ssn	pname	address	sid	gpa
1234545	John	216 Rosse	001	3.4
5423341	Bob	216 Rosse	002	1.3

If the two joined relations already have the same attribute name as join attribute, then renaming is not necessary.

(Semi) JOIN

Students(sid,sname,gpa)
People(ssn,pname,address)

- Notation: $R \ltimes S$
- $R \ltimes S = \pi_{A_1,A_2,\ldots A_n}(R \bowtie S)$
  - Where $A_1, A_2, \ldots A_n$ are attributes in R
- Example:
  - $People \ltimes Students$

ssn	pname	address	sid	gpa
1234545	John	216 Rosse	001	3.4
5423341	Bob	216 Rosse	002	1.3

SQL:

```
SELECT DISTINCT
  ssn,pname,address
FROM
  People, Students
WHERE
  pname = sname;
```

RA:

$$People \ltimes Students$$

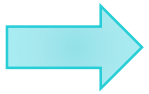
Exercise: Natural Join

- Given schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?
- Given R(A, B, C), S(D, E), what is R ⋈ S ?
- Given R(A, B), S(A, B), what is R ⋈ S ?

Example: Converting SFW Query -> RA

Students(sid,sname,gpa)
People(ssn,sname,address)

```
SELECT DISTINCT
  gpa,
  address
FROM Students S,
     People P
WHERE gpa > 3.5 AND
   sname = pname;
```



$$\Pi_{gpa,address}(\sigma_{gpa>3.5}(S \bowtie P))$$

How do we represent this query in RA?

DIVISION

Division $T \leftarrow R \div S$ can be expressed as a sequence of these operations:

- Example:
  - $T \leftarrow R \div S$

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

T

B
b1
b4

1 $T_1 \leftarrow \pi_B(R)$

T_1

B
b1
b2
b3
b4

2 $T_2 \leftarrow \pi_B((S \times T_1) - R)$

$S \times T_1$

A	B
a1	b1
a2	b1
a3	b1
a1	b2
a2	b2
a3	b2
a1	b3
a2	b3
a3	b3
a1	b4
a2	b4
a3	b4

$-$ R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

$=$ T_2

B
b2
b3

3 $T \leftarrow T_1 - T_2$

T

B
b1
b4

DIVISION

- Example:
  - *Retrieve the names of employees who work on all projects that 'John Smith' works on*

$$SMITH \leftarrow \sigma_{Fname='John' \textbf{ AND } Lname='Smith'}(EMPLOYEE)$$
$$SMITH\_PNOS \leftarrow \pi_{Pno}(WORKS\_ON \bowtie_{Essn=Ssn} SMITH)$$

$$SSN\_PNOS \leftarrow \pi_{Essn,\ Pno}(WORKS\_ON)$$

$$SSNS(Ssn) \leftarrow SSN\_PNOS \div SMITH\_PNOS$$
$$RESULT \leftarrow \pi_{Fname,\ Lname}(SSNS * EMPLOYEE)$$

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

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 $R_1 \bowtie_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$
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 $R_1 *_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$ 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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Exercise

- Retrieve the name and address of all employees who work for the 'Research' department.
- Retrieve the names of all employees in department 5 who work more than 10 hours per week on the ProductX project.
- List the names of all employees who have a dependent with the same first name as themselves.
- Find the names of all employees who are directly supervised by 'Franklin Wong'.
- Find the names of employees who work on *all* the projects controlled by department number 5.

Query Tree

$$\pi_{Pnumber,\ Dnum,\ Lname,\ Address,\ Bdate}(((\sigma_{Plocation='Stafford'}(PROJECT)) \bowtie_{Dnum=Dnumber}(DEPARTMENT)) \bowtie_{Mgr\_ssn=Ssn}(EMPLOYEE))$$

- Example:

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

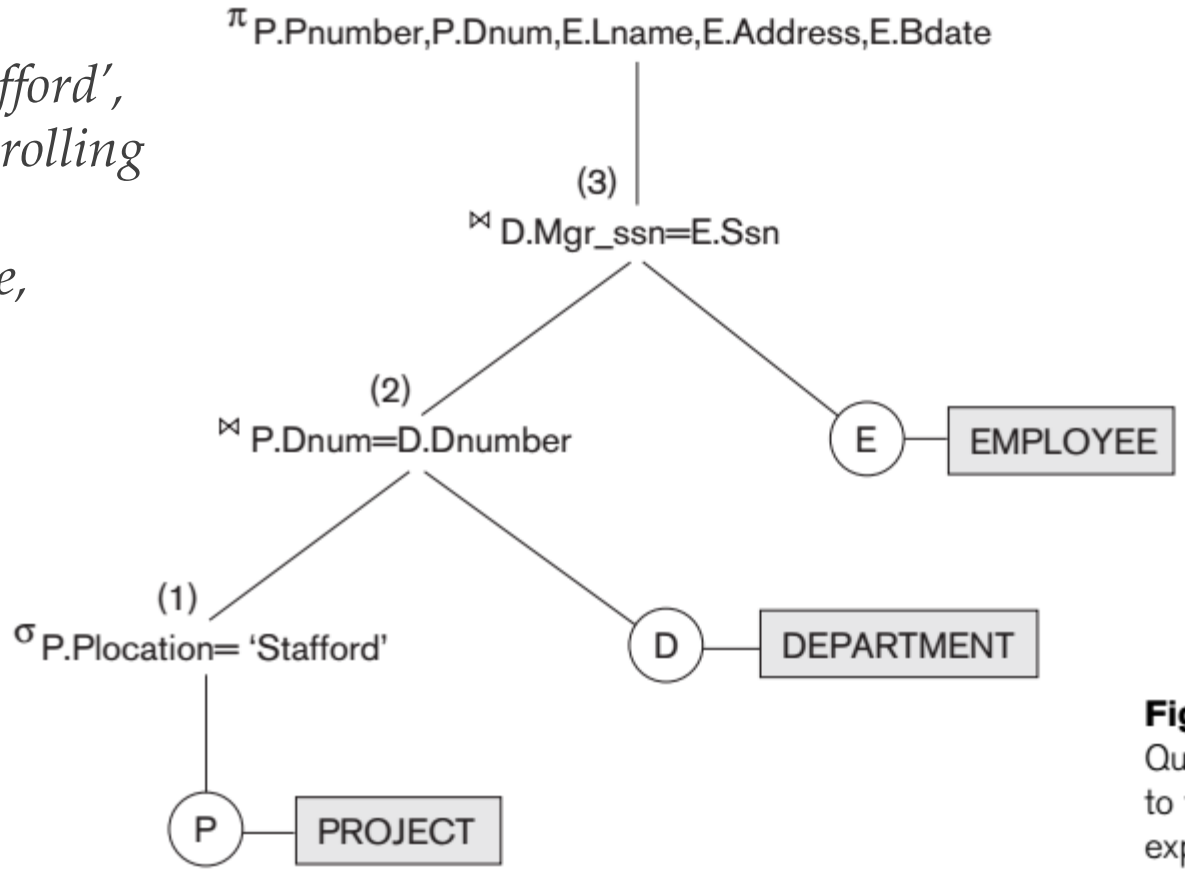
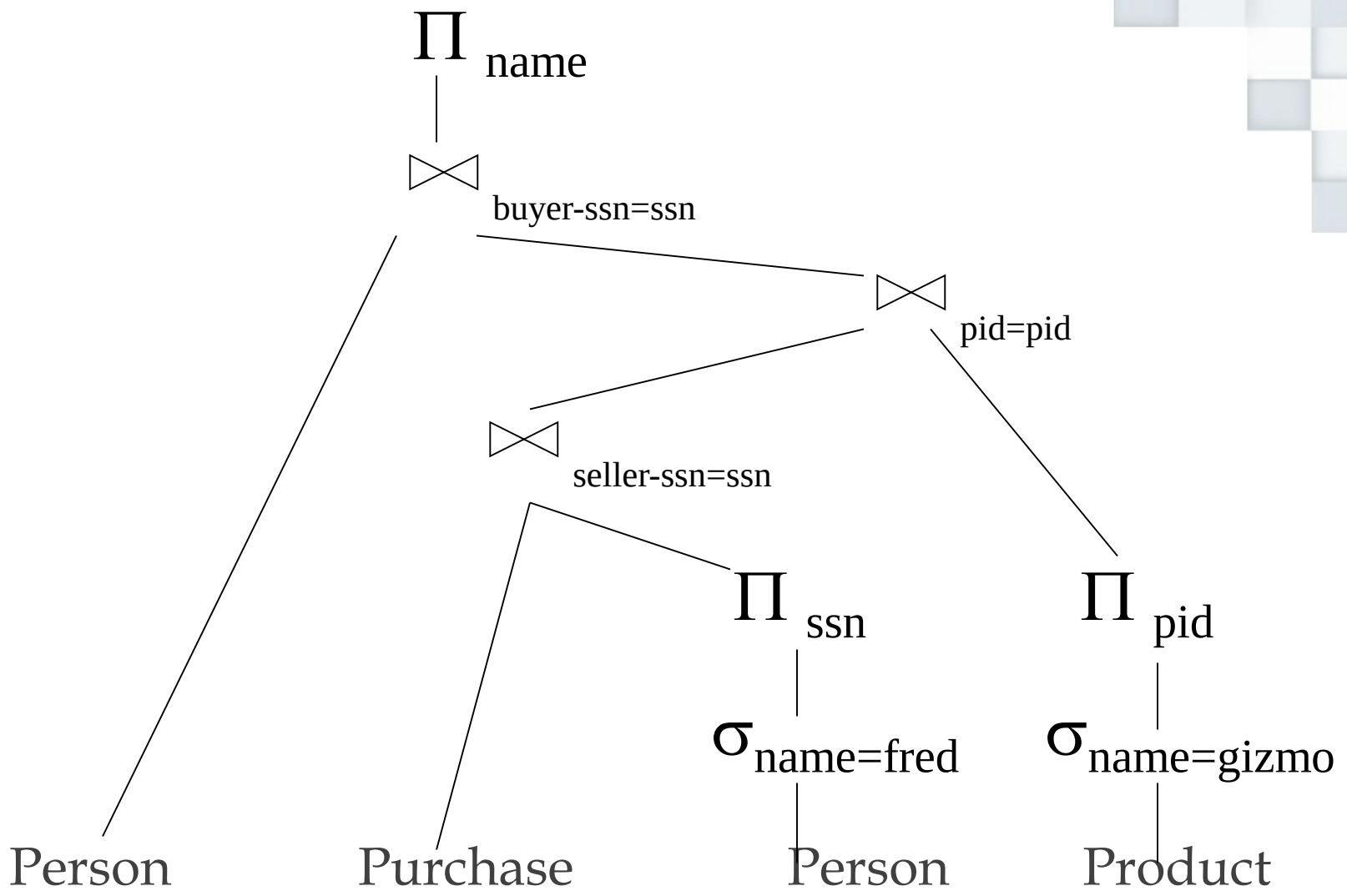


Figure 6.9
Query tree corresponding to the relational algebra expression for Q2.

Query Tree


Π name
buyer-ssn=ssn
pid=pid
seller-ssn=ssn
Π ssn
Π pid
σ name=fred
σ name=gizmo
Person
Purchase
Person
Product

Additional RA Operations

- Aggregate & Grouping
- Recursive Closure Relation
- Outer Join
- Outer Union

Aggregate & Grouping

- Notation: $<grouping\ attributes>\Im_{<function\ list>}(R)$
- Example:
  - *Retrieve each department number, the number of employees in the department, and their average salary.*
  - $\rho_{R(Dno,No\_of\_employees,Average\_sal)}\left(Dno\Im_{COUNT\ Ssn,AVERAGE\ Salary}(Employee)\right)$

Renaming relation and attributes

Group by Dno

R

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

Aggregate & Grouping

- If no grouping attributes are specified, then the functions are applied to all tuples in the relation.
- Example:
  - *Retrieve the number of employees and their average salary.*
  - $\rho_{R(No\_of\_employees, Average\_sal)} \left(\Im_{COUNT\ Ssn, AVERAGE\ Salary} (Employee) \right)$

R

Count_ssn	Average_salary
8	35125

Recursive Closure

- Applied to a recursive relationship between tuples of the same type.
- Example:
  - *Retrieve all supervisees (at all levels) of an employee whose name is 'James Borg'.*
  - We utilize a looping mechanism
    - (Level 1) *Retrieve all direct supervisees of an employee whose name is 'James Borg'.*
    - (Level 2) *Retrieve all supervisees of some employee who is directly supervised by 'James Borg'.*
    - ….
    - (Level n) …

Recursive Closure

- Example (cont'd):
  - (Level 1) *Retrieve all direct supervisees of an employee whose name is 'James Borg'.*

$$\text{BORG\_SSN} \leftarrow \pi_{\text{Ssn}}(\sigma_{\text{Fname='James' AND Lname='Borg'}}(\text{EMPLOYEE}))$$
$$\text{SUPERVISION}(\text{Ssn1}, \text{Ssn2}) \leftarrow \pi_{\text{Ssn,Super\_ssn}}(\text{EMPLOYEE})$$
$$\text{RESULT1}(\text{Ssn}) \leftarrow \pi_{\text{Ssn1}}(\text{SUPERVISION} \bowtie_{\text{Ssn2=Ssn}} \text{BORG\_SSN})$$

  - (Level 2) *Retrieve all supervisees of some employee who is directly supervised by 'James Borg'.*

$$\text{RESULT2}(\text{Ssn}) \leftarrow \pi_{\text{Ssn1}}(\text{SUPERVISION} \bowtie_{\text{Ssn2=Ssn}} \text{RESULT1})$$

  - To get both set of employees supervised at level 1 and 2 by James Borg:

$$\text{RESULT} \leftarrow \text{RESULT2} \cup \text{RESULT1}$$

Recursive Closure

SUPERVISION

(Borg's Ssn is 888665555)

(Ssn)	(Super_ssn)
Ssn1	Ssn2
123456789	333445555
333445555	888665555
999887777	987654321
987654321	888665555
666884444	333445555
453453453	333445555
987987987	987654321
888665555	null

RESULT1

Ssn
333445555
987654321

(Supervised by Borg)

RESULT2

Ssn
123456789
999887777
666884444
453453453
987987987

(Supervised by Borg's subordinates)

RESULT

Ssn
123456789
999887777
666884444
453453453
987987987
333445555
987654321

(RESULT1 ∪ RESULT2)

Outer JOIN

- Notation: $R_1 \ltimes R_2$ (left) or $R_1 \rtimes R_2$ (right) or $R_1 \ltimes\!\rtimes R_2$ (full)
- Example:
  - Left Outer JOIN

$$TEMP \leftarrow (EMPLOYEE \ltimes_{Ssn=Mgr\_ssn} DEPARTMENT)$$

$$RESULT \leftarrow \pi_{Fname,\ Minit,\ Lname,\ Dname}(TEMP)$$

RESULT

Fname	Minit	Lname	Dname
John	B	Smith	NULL
Franklin	T	Wong	Research
Alicia	J	Zelaya	NULL
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	NULL
Joyce	A	English	NULL
Ahmad	V	Jabbar	NULL
James	E	Borg	Headquarters

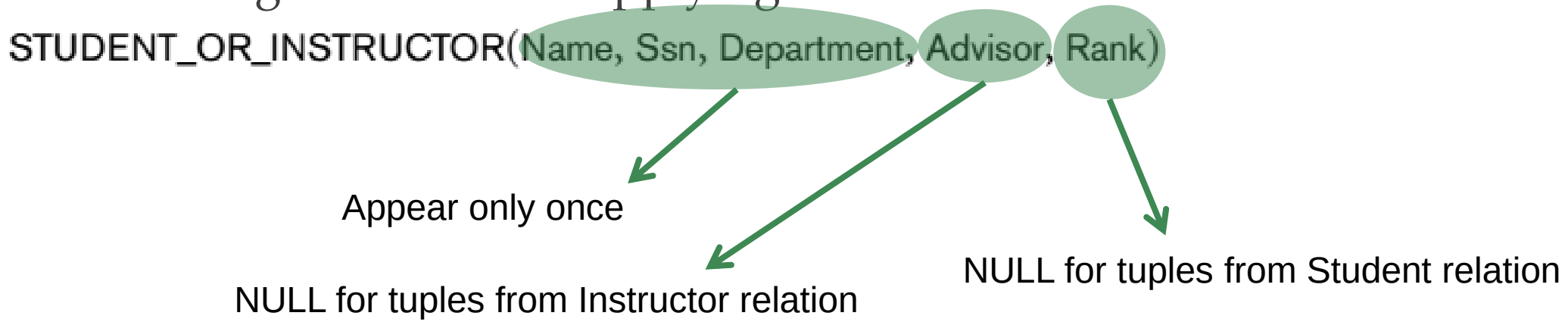
Outer UNION

- Take the union of tuples from two relations if the relations are **not union compatible.**
- Takes **UNION** of tuples in two relations R(X,Y) and S(X,Z) that are **partially compatible**, meaning that only some of their attributes, say X, are union compatible.
- The result relation T(X,Y,Z), the attributes that are union compatible are represented only once in the result, and those which are not union compatible from either relation are also kept in the result.

Outer UNION

- Example:
  - Student(Name, Ssn, Department, Advisor)
  - Instructor(Name, Ssn, Department, Rank)

  - The resulting relation from applying Outer Union:



Exercise

- For each department, retrieve the department name and the average salary of all employees working in that department.
- Retrieve the average salary of all female employees.
- Show the result of following operation:

a. $T1 \bowtie_{T1.P = T2.A} T2$

b. $T1 \bowtie_{T1.Q = T2.B} T2$

c. $T1 \bowtie_{T1.P = T2.A} T2$

d. $T1 \bowtie_{T1.Q = T2.B} T2$

e. $T1 \cup T2$

f. $T1 \bowtie_{(T1.P = T2.A \text{ AND } T1.R = T2.C)} T2$

TABLE T1

P	Q	R
10	a	5
15	b	8
25	a	6

TABLE T2

A	B	C
10	b	6
25	c	3
10	b	5