Relational Algebra

Relational Algebra

- Relational algebra is often considered to be an integral part of the relational data model.
- Its operation can be divide into two groups:
    1. Set operations from mathematical set theory; these are applicable because each relation defined to be set of tuple In the formal relational model.

       Union, intersection, set Difference and cartesian Product

    2. Consist of operation developed specifically for relational databases

       Select, project, join

Relational Algebra

- Theoretical way of manipulating table contents using relational operators
- **Relvar**: Variable that holds a relation
  - Heading contains the names of the attributes and the body contains the relation
- Relational operators have the property of closure
  - **Closure**: Use of relational algebra operators on existing relations produces new relations

Relational Set Operators

Select

- Unary operator that yields a horizontal subset of a table

Project

- Unary operator that yields a vertical subset of a table

Union

- Combines all rows from two tables, excluding duplicate rows
- **Union-compatible**: Tables share the same number of columns, and their corresponding columns share compatible domains

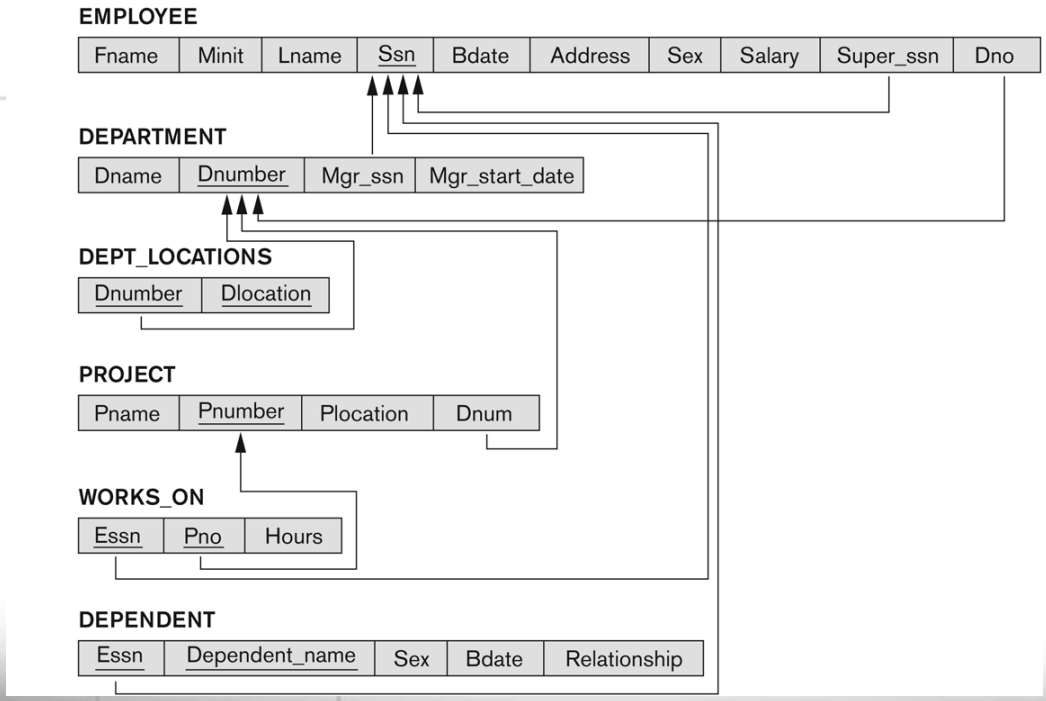
Intersect

- Yields only the rows that appear in both tables
- Tables must be union-compatible to yield valid results

Unary Relational Operations

Select  **PROJECT**

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.

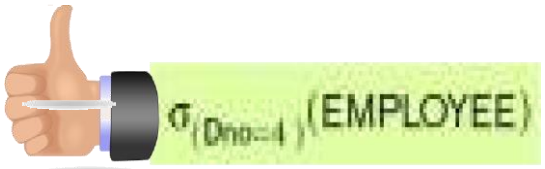


Unary = Applied to a single relation

SELECT Example

Query

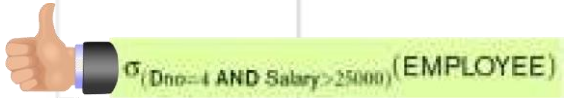
Find the list of employees from department 4 ?



SELECT Example

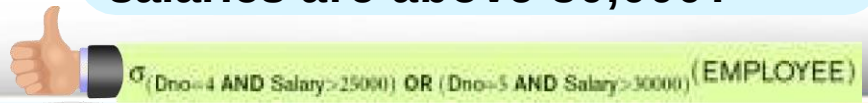
Query

Find the list of employees from department 4 whose salaries is above 25,000?



Query

Find the list of employees from department 4 whose salaries is above 25,000 or from department 5 whose salaries are above 30,000?



SELECT Example

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

SELECT only PRICE less than $2.00 yields

$\sigma_{Price < 2.00}(Product)$

P_CODE	P_DESCRIPT	PRICE
213345	9v battery	1.92
254467	100W bulb	1.47

SELECT only P_CODE = 311452 yields

$\sigma_{P\_CODE = 311452}(Product)$

P_CODE	P_DESCRIPT	PRICE
311452	Powerdrill	34.99

SELECT Example

- What's going on?
- <selection condition> applied independently to each individual tuple t in R
  - If condition evaluates to TRUE, tuple selected
- Boolean conditions AND, OR, and NOT

Selectivity

- Fraction of tuples selected by a selection condition

- SELECT operation is commutative
- Cascade SELECT operations into a single operation with AND condition

Unary Relational Operations: SELECT (continued)

- SELECT Operation Properties
  - The SELECT operation $\sigma_{<\text{selection condition}>}(R)$ produces a relation S that has the same schema (same attributes) as R
  - SELECT σ is commutative:
    - $\sigma_{<\text{condition1}>}(\sigma_{<\text{condition2}>}(R)) = \sigma_{<\text{condition2}>}(\sigma_{<\text{condition1}>}(R))$
  - Because of commutativity property, a cascade (sequence) of SELECT operations may be applied in any order:
    - $\sigma_{<\text{cond1}>}(\sigma_{<\text{cond2}>}(\sigma_{<\text{cond3}>}(R)) = \sigma_{<\text{cond2}>}(\sigma_{<\text{cond3}>}(\sigma_{<\text{cond1}>}(R)))$
  - A cascade of SELECT operations may be replaced by a single selection with a conjunction of all the conditions:
    - $\sigma_{<\text{cond1}>}(\sigma_{<\text{cond2}>}(\sigma_{<\text{cond3}>}(R)) = \sigma_{<\text{cond1}> \text{ AND } <\text{cond2}> \text{ AND } <\text{cond3}>}(R))$
  - The number of tuples in the result of a SELECT is less than (or equal to) the number of tuples in the input relation R

The PROJECT Operation

Selects columns from table and discards the other columns:

$$\pi_{<\text{attribute list}>}(R)$$

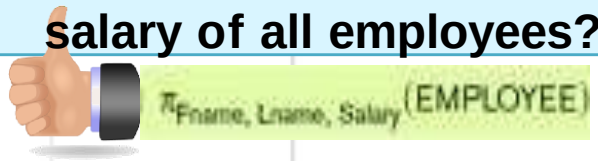
Degree: Number of attributes in <attribute list>

Duplicate elimination: Result of PROJECT operation is a set of distinct tuples

The PROJECT Example

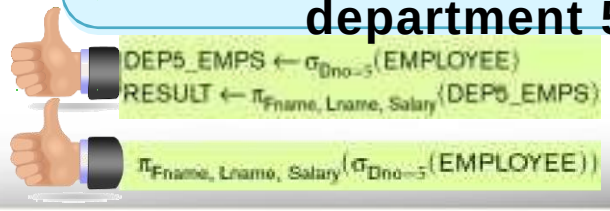
Query

Find/Retrieve the list of first name, last name and salary of all employees?



Query

Find the list of first name, last name and salary of all employees who work in department 5?



The PROJECT Example

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

PROJECT PRICE yields

$\pi_{Price}(Product)$

New table

PRICE
5.26
25.15
10.99
1.92
1.47
34.99

PROJECT P_DESCRIPT and PRICE yields

$\pi_{P\_Description,Price}(Product)$

P_DESCRIPT	PRICE
Flashlight	5.26
Lamp	25.15
Box Fan	10.99
9v battery	1.92
100W bulb	1.47
Powerdrill	34.99

PROJECT P_CODE and PRICE yields

$\pi_{P\_Code,Price}(Project)$

P_CODE	PRICE
123456	5.26
123457	25.15
123458	10.99
213345	1.92
254467	1.47
311452	34.99

Unary Relational Operations: PROJECT

- PROJECT Operation Properties
  - The number of tuples in the result of projection $\pi_{<list>}(R)$ is always less or equal to the number of tuples in R
    - If the list of attributes includes a *key* of R, then the number of tuples in the result of PROJECT is *equal* to the number of tuples in R
  - PROJECT is *not* commutative
    - $\pi_{<list1>}(\pi_{<list2>}(R)) = \pi_{<list1>}(R)$ as long as <list2> contains the attributes in <list1>

Single expression versus sequence of relational operations (Example)

- To retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a select and a project operation
- We can write a *single relational algebra expression* as follows:
  - $\pi_{FNAME, LNAME, SALARY}(\sigma_{DNO=5}(EMPLOYEE))$
- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
  - DEP5_EMPS $\leftarrow \sigma_{DNO=5}(EMPLOYEE)$
  - RESULT $\leftarrow \pi_{FNAME, LNAME, SALARY}$ (DEP5_EMPS)

Unary Relational Operations: **RENAME**

- The RENAME operator is denoted by ρ (rho)
- In some cases, we may want to *rename* the attributes of a relation or the relation name or both
  - Useful when a query requires multiple operations
  - Necessary in some cases

Unary Relational Operations: RENAME

- The general RENAME operation ρ can be expressed by any of the following forms:
  - $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes both:
    - the relation name to S, *and*
    - the column (attribute) names to B1, B1, …..Bn
  - $\rho_S(R)$ changes:
    - the *relation name* only to S
  - $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes:
    - the *column (attribute) names* only to B1, B1, …..Bn

Unary Relational Operations: RENAME

- For convenience, we also use a *shorthand* for renaming attributes in an intermediate relation:
  - If we write:
    - RESULT $\leftarrow \pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS)
    - RESULT will have the *same attribute names* as DEP5_EMPS (same attributes as EMPLOYEE)
  - If we write:
    - RESULT (F, M, L, S, B, A, SX, SAL, SU, DNO)$\leftarrow \rho_{\text{RESULT}(\text{F.M.L.S.B,A,SX,SAL,SU, DNO})}$(DEP5_EMPS)
    - The 10 attributes of DEP5_EMPS are *renamed* to F, M, L, S, B, A, SX, SAL, SU, DNO, respectively

  Note: the $\leftarrow$ symbol is an assignment operator

Example of applying multiple operations and RENAME

Fname	Lname	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

$\pi_{FNAME, LNAME, SALARY}(\sigma_{DNO=5}(EMPLOYEE))$

First_name	Last_name	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

$R(First\_name, Last\_name, Salary) \leftarrow \pi_{FNAME, LNAME, SALARY}(Temp)$

TEMP

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston,TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston,TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble,TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

$Temp \leftarrow \sigma_{DNO=5}(EMPLOYEE)$

Binary Operations

Relational Algebra Operations from Set Theory: **UNION**

- UNION Operation
  - Binary operation, denoted by ∪
  - The result of R ∪ S, is a relation that includes all tuples that are either in R or in S or in both R and S
  - Duplicate tuples are eliminated
  - The two operand relations R and S must be “type compatible” (or UNION compatible)
    - R and S must have same number of attributes
    - Each pair of corresponding attributes must be type compatible (have same or compatible domains)

Relational Algebra Operations from Set Theory: UNION

- **Example:**
  - To retrieve the social security numbers of all employees who either *work in department 5* or *directly supervise an employee who works in department 5*
  - The union operation produces the tuples that are in either RESULT1 or RESULT2 or both

Result of the UNION operation RESULT ← RESULT1 ∪ RESULT2.

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

RESULT2(SSN) ← $\pi_{SUPERSSN}$(DEP5_EMPS)

DEP5_EMPS ← $\sigma_{DNO=5}$ (EMPLOYEE)
RESULT1 ← π_{SSN}(DEP5_EMPS)

RESULT ← RESULT1 ∪ RESULT2

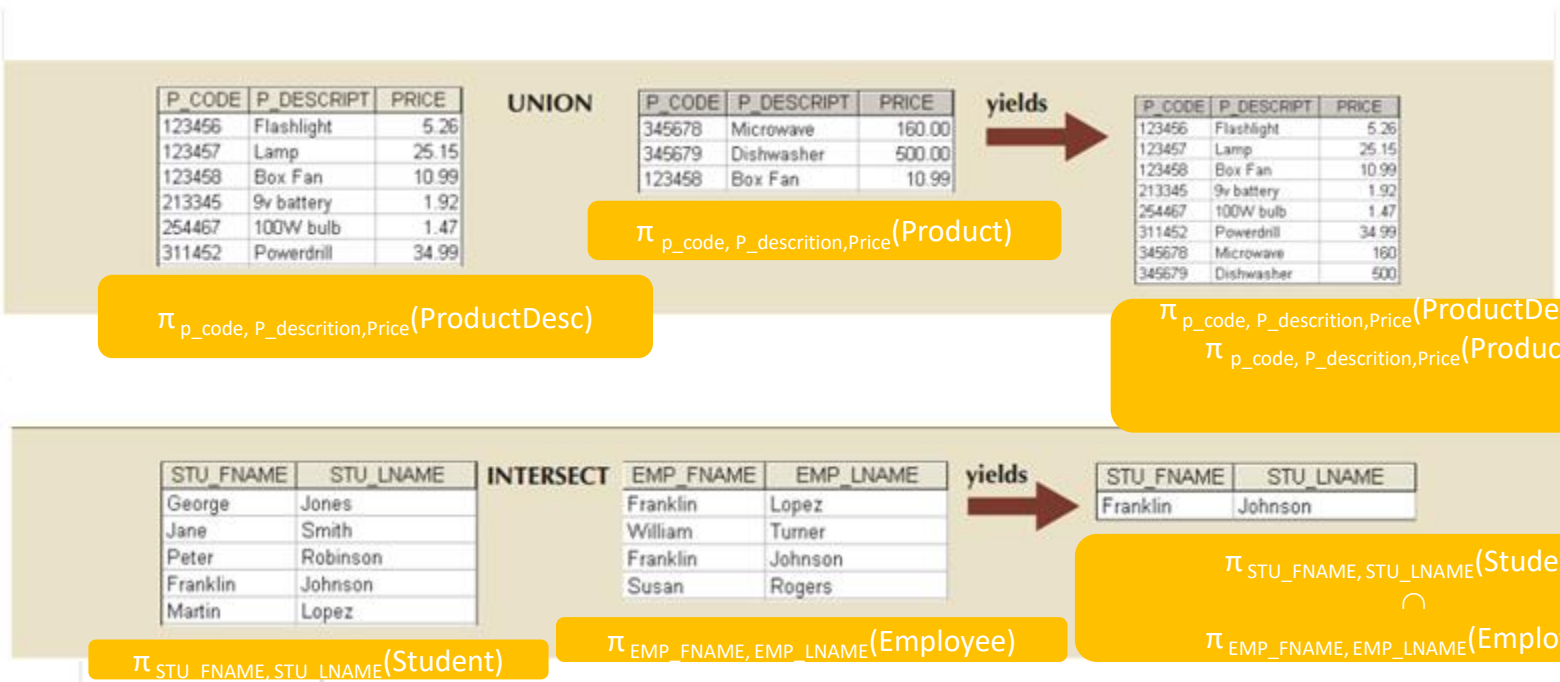
Relational Algebra Operations from Set Theory

- Type Compatibility of operands is required for the binary set operation UNION $\cup$, (also for INTERSECTION $\cap$, and SET DIFFERENCE –)
- R1(A1, A2, ..., An) and R2(B1, B2, ..., Bn) are type compatible if:
  - they have the same number of attributes, and
  - the domains of corresponding attributes are type compatible (i.e. dom(Ai)=dom(Bi) for i=1, 2, ..., n).
- The resulting relation for R1$\cup$R2 (also for R1$\cap$R2, or R1–R2)has the same attribute names as the *first* operand relation R1 (by convention)

Relational Algebra Operations from Set Theory: **INTERSECTION**

- INTERSECTION is denoted by $\cap$
- The result of the operation R $\cap$ S, is a relation that includes all tuples that are in both R and S
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be “type compatible”

Union and Intersect



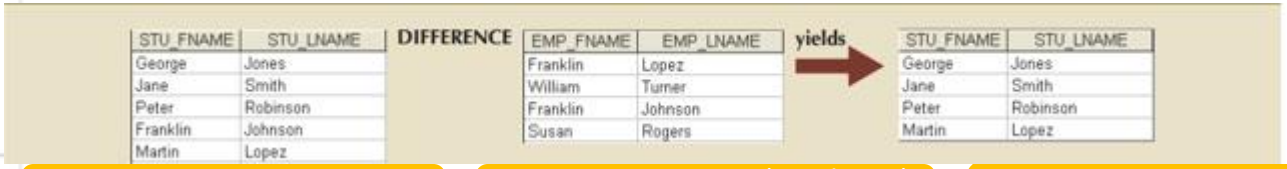
Relational Algebra Operations from Set Theory: **SET DIFFERENCE**

- SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by –
- The result of R – S, is a relation that includes all tuples that are in R but not in S
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be “type compatible”

Relational Set Operators

- **Difference**
  - Yields all rows in one table that are not found in the other table
  - Tables must be union-compatible to yield valid results

Difference



STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

DIFFERENCE

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Martin	Lopez

$\pi_{STU\_FNAME, STU\_LNAME}(Student)$

$\pi_{EMP\_FNAME, EMP\_LNAME}(Employee)$

$\pi_{STU\_FNAME, STU\_LNAME}(Student) - \pi_{EMP\_FNAME, EMP\_LNAME}(Employee)$

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R DIFFERENCE S

B	2
F	4
E	5

S DIFFERENCE R

C	2
E	4

Some properties of UNION, INTERSECT, and DIFFERENCE

- Notice that both union and intersection are *commutative* operations; that is
  - $R \cup S = S \cup R$, and $R \cap S = S \cap R$
- Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative* operations; that is
  - $R \cup (S \cup T) = (R \cup S) \cup T$
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- The minus operation is not commutative; that is, in general
  - $R - S \neq S - R$

(a) 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

(b)

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

Student U Instructor

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

Student ∩ Instructor

(d)

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

Student - Instructor

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

Instructor - Student

Relational Algebra Operations from Set Theory: **CARTESIAN PRODUCT**

- CARTESIAN (or CROSS) PRODUCT Operation
  - This operation is used to combine tuples from two relations in a combinatorial fashion.
  - Denoted by R(A1, A2, . . ., An) x S(B1, B2, . . ., Bm)
  - Result is a relation Q with degree n + m attributes:
    - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
  - The resulting relation state has one tuple for each combination of tuples—one from R and one from S.
  - Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then R x S will have $n_R * n_S$ tuples.
  - The two operands do NOT have to be "type compatible"
- Generally, CROSS PRODUCT is not a meaningful operation
  - Can become meaningful when followed by other operations

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

$$\text{FEMALE\_EMPS} \leftarrow \sigma_{\text{SEX='F'}}(\text{EMPLOYEE})$$

EMPNAMES

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

$$\text{EMPNAMES} \leftarrow \pi_{\text{FNAME, LNAME, SSN}}(\text{FEMALE\_EMPS})$$

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

EMP_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	...
Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	...
Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	...
Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	...
Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	...
Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	...
Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	...
Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	...
Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	...
Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...
Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	...
Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	...
Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	...
Joyce	English	453453453	333445555	Alice	F	1986-04-05	...
Joyce	English	453453453	333445555	Theodore	M	1983-10-25	...
Joyce	English	453453453	333445555	Joy	F	1958-05-03	...
Joyce	English	453453453	987654321	Abner	M	1942-02-28	...
Joyce	English	453453453	123456789	Michael	M	1988-01-04	...
Joyce	English	453453453	123456789	Alice	F	1988-12-30	...
Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	...

$$\text{EMP\_DEPENDENTS} \leftarrow \text{EMPNAMES} \times \text{DEPENDENT}$$

Meaningful Representation

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	...
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	...

$$\text{ACTUAL\_DEPS} \leftarrow \sigma_{\text{SSN=ESSN}}(\text{EMP\_DEPENDENTS})$$

RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

$$\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, DEPENDENT\_NAME}}(\text{ACTUAL\_DEPS})$$

Product

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

result1<-$\pi_{P\_Code,\ P\text{-}Descript,\ Price}$(Product)

PRODUCT

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

result2<-$\pi_{Store,\ Aisle,\ Shelf}$(Store)

yields

P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	5.26	23	W	5
123456	Flashlight	5.26	24	K	9
123456	Flashlight	5.26	25	Z	6
123457	Lamp	25.15	23	W	5
123457	Lamp	25.15	24	K	9
123457	Lamp	25.15	25	Z	6
123458	Box Fan	10.99	23	W	5
123458	Box Fan	10.99	24	K	9
123458	Box Fan	10.99	25	Z	6
213345	9v battery	1.92	23	W	5
213345	9v battery	1.92	24	K	9
213345	9v battery	1.92	25	Z	6
311452	Powerdrill	34.99	23	W	5
311452	Powerdrill	34.99	24	K	9
311452	Powerdrill	34.99	25	Z	6
254467	100W bulb	1.47	23	W	5
254467	100W bulb	1.47	24	K	9
254467	100W bulb	1.47	25	Z	6

result1 x result2

Binary Relational Operators

Binary Relational Operations: JOIN

- JOIN Operation (denoted by $\bowtie$)
  - The sequence of CARTESIAN PRODECT followed by SELECT is used quite commonly to identify and select related tuples from two relations
  - A special operation, called JOIN combines this sequence into a single operation
  - This operation is very important for any relational database with more than a single relation, because it allows us *combine related tuples* from various relations
  - The general form of a join operation on two relations R(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is:

$$R \bowtie_{\text{<join condition>}} S$$

  - where R and S can be any relations that result from general *relational algebra expressions*.

Example: Suppose that we want to retrieve the name of the manager of each department.

To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.

DEPT_MGR

Dname	Dnumber	Mgr_ssn	...	Fname	Minit	Lname	Ssn	...
Research	5	333445555	...	Franklin	T	Wong	333445555	...
Administration	4	987654321	...	Jennifer	S	Wallace	987654321	...
Headquarters	1	888665555	...	James	E	Borg	888665555	...

$$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRSSN=SSN}} \text{EMPLOYEE}$$

- MGRSSN=SSN is the join condition
- Combines each department record with the employee who manages the department
- The join condition can also be specified as DEPARTMENT.MGRSSN= EMPLOYEE.SSN

$$EMP\_DEPENDENTS \leftarrow EMPNAMES \times DEPENDENT$$

$$ACTUAL\_DEPS \leftarrow \sigma_{SSN=ESSN}(EMP\_DEPENDENTS)$$

Can be written as

$$ACTUAL\_DEPENDENTS \leftarrow EMPNAMES \bowtie_{SSN=ESSN} DEPENDENT$$

$$EMPNAMES \leftarrow \pi_{FNAME, LNAME, SSN}(FEMALE\_EMPS)$$

Some properties of JOIN

- Consider the following JOIN operation:
  - R(A1, A2, . . ., An) $\bowtie_{R.Ai=S.Bj}$ S(B1, B2, . . ., Bm)
  - Result is a relation Q with degree n + m attributes:

    Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
  - The resulting relation state has one tuple for each combination of tuples r from R and s from S, but *only if they satisfy the join condition* r[Ai]=s[Bj]
  - Hence, if R has n_R tuples, and S has n_S tuples, then the join result will generally have *less than* $n_R * n_S$ tuples.
  - Only related tuples (based on the join condition) will appear in the result

Types of Joins

- **Natural join**: Links tables by selecting only the rows with common values in their common attributes
  - **Join columns**: Common columns
- **Equijoin**: Links tables based on an equality condition that compares specified columns of each table
- **Theta join**: Extension of natural join, denoted by adding a theta subscript after the JOIN symbol

Theta Join

- The general case of JOIN operation is called a Theta-join:

- The join condition is called *theta*
- *Theta* can be any general Boolean expression on the attributes of R and S; for example:

R.Ai < S.Bj AND (R.Ak = S.Bl OR R.Ap < S.Bq)

- Most join conditions involve one or more equality conditions "AND"ed together; for example:

R.Ai = S.Bj AND R.Ak = S.Bl AND R.Ap = S.Bq

```
SELECT e.emp_id, e.name, e.salary, d.dept_name, d.budget FROM
Employees e JOIN Departments d ON e.dept_id = d.dept_id WHERE e.salary
> d.budget;
```

EQUIJOIN

- The most common use of join involves join conditions with *equality comparisons* only
- Such a join, where the only comparison operator used is =, is called an EQUIJOIN.
  - In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have identical values in every tuple.

NATURAL JOIN

- Another variation of JOIN called NATURAL JOIN — denoted by * — was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition.
  - because one of each pair of attributes with identical values is superfluous
- The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, *have the same name* in both relations
- If this is not the case, a renaming operation is applied first.

NATURAL JOIN

- Example: To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write:

  DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS

- Only attribute with the same name is DNUMBER
- An implicit join condition is created based on this attribute:

  DEPARTMENT.DNUMBER=DEPT_LOCATIONS.DNUMBER

Example of NATURAL JOIN operation

PROJ_DEPT

Pname	Pnumber	Plocation	Dnum	Dname	Mgr_ssn	Mgr_start_date
ProductX	1	Bellaire	5	Research	333445555	1988-05-22
ProductY	2	Sugarland	5	Research	333445555	1988-05-22
ProductZ	3	Houston	5	Research	333445555	1988-05-22
Computerization	10	Stafford	4	Administration	987654321	1995-01-01
Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

proj_dept ← project * dept.

DEPT_LOCS

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Location
Headquarters	1	888665555	1981-06-19	Houston
Administration	4	987654321	1995-01-01	Stafford
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston

dept_locs ← department * dept_locations.

Natural and theta join

t1

a	b
a1	b1
a2	b2
a3	b3

t2

b	c
b1	c1
b2	c2
b4	c4

t1 ⋈ t2

a	b	c
a1	b1	c1
a2	b2	c2

course

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

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

name
C1
C3

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

name
C2
C1

Complete Set of Relational Operations

- The set of operations including SELECT σ, PROJECT π , UNION $\cup$, DIFFERENCE – , RENAME ρ, and CARTESIAN PRODUCT X is called a *complete set* because any other relational algebra expression can be expressed by a combination of these five operations.
- For example:

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

$$R \bowtie_{\text{<join condition>}} S = \sigma_{\text{<join condition>}} (R \times S)$$

Binary Relational Operations: DIVISION

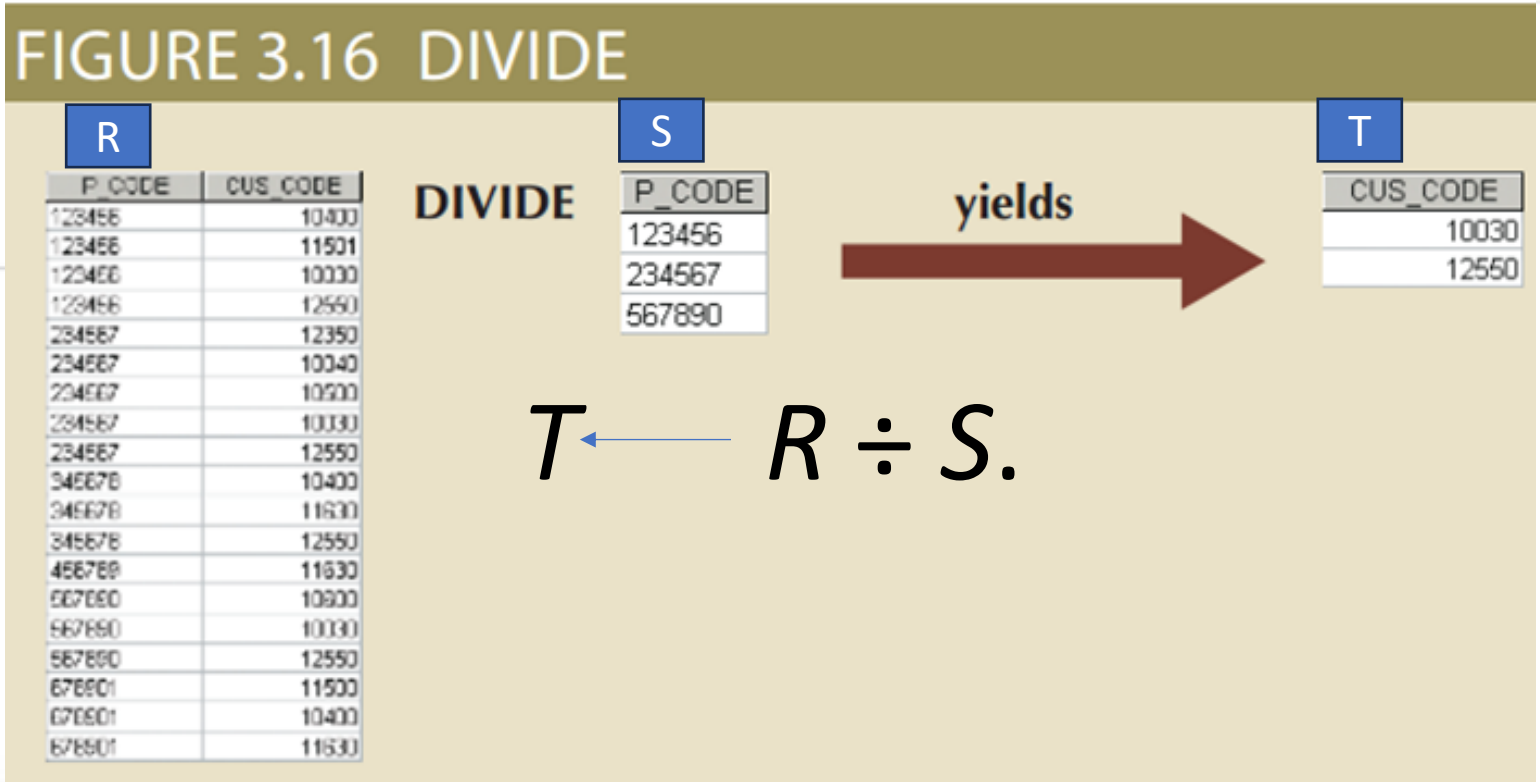
•Uses one 2-column table as the dividend and one single-column table as the divisor

•Output is a single column that contains all values from the second column of the dividend that are associated with every row in the divisor

- The division operation is applied to two relations
- R(Z) ÷ S(X), where X subset Z. Let Y = Z - X (and hence Z = X ∪ Y); that is, let Y be the set of attributes of R that are not attributes of S.
- The result of DIVISION is a relation T(Y) that includes a tuple t if tuples t_R appear in R with t_R [Y] = t, and with

  t_R [X] = t_s *for every tuple* t_s in S.
- For a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with *every* tuple in S.

Figure 3.16 - Divide



Illustration

completed

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

DBProject

task
Database1
Database2

completed / DBProject

student
Fred
Sara

Completed ← *Complete ÷ DBProject*.

If **DBProject** contains all the tasks of the Database project, then the result of the division above contains exactly the students who have completed both of the tasks in the Database project.

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

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5

SMITH ← $\sigma_{\text{Fname='John' AND Lname='Smith'}}(\text{EMPLOYEE})$

SMITH_PNOS

Pno
1
2

SMITH_PNOS ← $\pi_{\text{Pno}}(\text{WORKS\_ON} \bowtie_{\text{Essn=Ssn}} \text{SMITH})$

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

SSN_PNOS ← $\pi_{\text{ESSn,Pno}}(\text{WORKS\_ON})$

SSNS

Ssn
123456789
453453453

SSNS(Ssn) ← SSN_PNOS ÷ SMITH_PNOS

Fname	Lname
John	Smith
Joyce	English

RESULT ← $\pi_{\text{Fname, Lname}}(\text{SSNS} * \text{EMPLOYEE})$

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

Operations of Relational Algebra

Table 8.1 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 $R_1 * R_2$

Operations of Relational Algebra (continued)

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

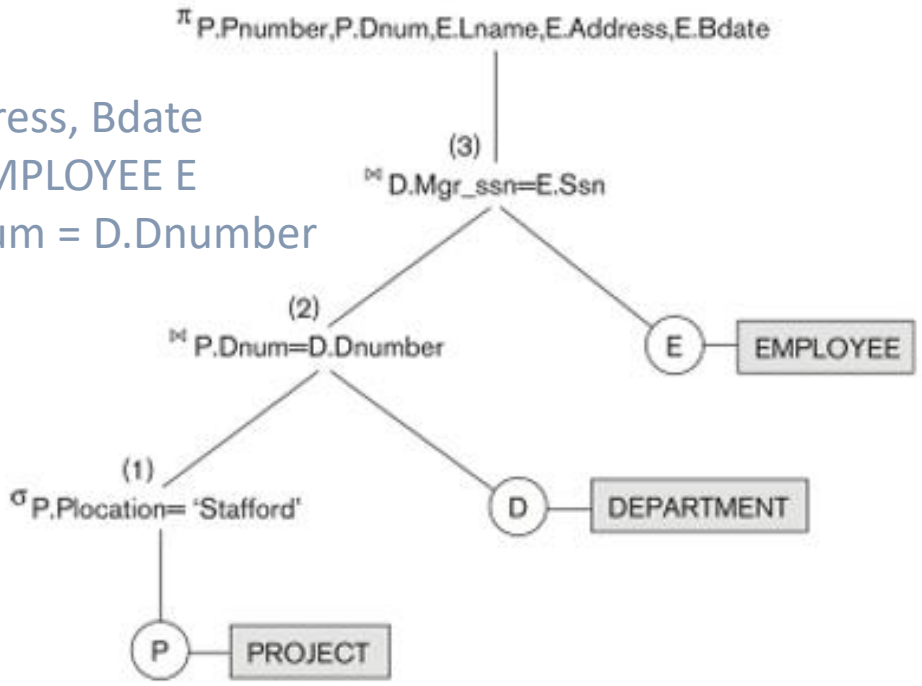
Query Tree Notation

- 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.

Example of Query Tree

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.

SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM PROJECT P, DEPARTMENT D, EMPLOYEE E
WHERE D.Mgr_ssn = E.Ssn **AND P.**Dnum = D.Dnumber
And Plocation = 'Stafford'



Additional Relational Operations: Aggregate Functions and Grouping

- A type of request that cannot be expressed in the basic relational algebra is to specify mathematical **aggregate functions** on collections of values from the database.
- Examples of such functions include retrieving the average or total salary of all employees or the total number of employee tuples.
  - These functions are used in simple statistical queries that summarize information from the database tuples.
- Common functions applied to collections of numeric values include SUM, AVERAGE, MAXIMUM, and MINIMUM.
- The COUNT function is used for counting tuples or values.

Aggregate Function Operation

- Use of the Aggregate Functional operation $\mathcal{F}$
  - $\mathfrak{I}_{\text{MAX Salary}}$ (EMPLOYEE) retrieves the maximum salary value from the EMPLOYEE relation
  - $\mathfrak{I}_{\text{MIN Salary}}$ (EMPLOYEE) retrieves the minimum Salary value from the EMPLOYEE relation
  - $\mathfrak{I}_{\text{SUM Salary}}$ (EMPLOYEE) retrieves the sum of the Salary from the EMPLOYEE relation
  - $\mathfrak{I}_{\text{COUNT SSN, AVERAGE Salary}}$ (EMPLOYEE) computes the count (number) of employees and their average salary
    - Note: count just counts the number of rows, without removing duplicates

Using Grouping with Aggregation

- Example: For each department, retrieve the DNO, COUNT SSN, and AVERAGE SALARY
- A variation of aggregate operation $\mathcal{F}$ allows this:
  - Grouping attribute placed to left of symbol
  - Aggregate functions to right of symbol

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

Dno $\Im$ COUNT Ssn, AVERAGE Salary (EMPLOYEE)).

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

ρ_R(Dno, No_of_employees, Average_sal)(Dno $\Im$ COUNT Ssn, AVERAGE Salary (EMPLOYEE)).

Count_ssn	Average_salary
8	35125

$\Im$ COUNT Ssn, AVERAGE Salary(EMPLOYEE).

Results of GROUP BY and HAVING (in SQL).

Fname	Minit	Lname	Ssn	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

Additional Relational Operations (**Recursive Closure Operations**)

- An example of a recursive operation is to retrieve all SUPERVISEES of an EMPLOYEE **e** at all levels — that is, all EMPLOYEE **e'** directly supervised by **e**; all employees **e''** directly supervised by each employee **e'**; all employees **e'''** directly supervised by each employee **e''**; and so on.

Additional Relational Operations (continued)

- Although it is possible to retrieve employees at each level and then take their union, we cannot, in general, specify a query such as “retrieve the supervisees of ‘James Borg’ at all levels” without utilizing a looping mechanism.

A two-level recursive query.

Ssn1	Ssn2
123456789	333445555
333445555	888665555
999887777	987654321
987654321	888665555
666884444	333445555
453453453	333445555
987987987	987654321
888665555	null

Ssn
888665555

BORG_SSN $\leftarrow \pi_{Ssn}(\sigma_{Fname='James'\ \text{AND}\ Lname='Borg'}(EMPLOYEE))$

SUPERVISION(Ssn1, Ssn2) $\leftarrow \pi_{Ssn,Super\_ssn}(EMPLOYEE)$

RESULT1

Ssn
333445555
987654321

RESULT1(Ssn) $\leftarrow \pi_{Ssn1}(SUPERVISION \bowtie_{Ssn2=Ssn} BORG\_SSN)$

RESULT2

Ssn
123456789
999887777
666884444
453453453
987987987

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

RESULT $\leftarrow$ RESULT2 $\cup$ RESULT1

RESULT

Ssn
123456789
999887777
666884444
453453453
987987987

Additional Relational Operations (continued)

- The OUTER JOIN Operation
  - In NATURAL JOIN and EQUIJOIN, tuples without a *matching* (or *related*) tuple are eliminated from the join result
    - Tuples with null in the join attributes are also eliminated
    - This amounts to loss of information.
  - A set of operations, called OUTER joins, can be used when we want to keep all the tuples in R, or all those in S, or all those in both relations in the result of the join, regardless of whether or not they have matching tuples in the other relation.

Additional Relational Operations (continued)

- The left outer join operation keeps every tuple in the first or left relation R in R ⟕ S; if no matching tuple is found in S, then the attributes of S in the join result are filled or "padded" with null values.
- A similar operation, right outer join, keeps every tuple in the second or right relation S in the result of R ⟖ S.
- A third operation, full outer join, denoted by ⟗ keeps all tuples in both the left and the right relations when no matching tuples are found, padding them with null values as needed.

LEFT OUTER JOIN operation.

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

$$TEMP \leftarrow (EMPLOYEE \overset{}{\bowtie}_{Ssn=Mgr\_ssn} DEPARTMENT)$$

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

Additional Relational Operations (continued)

- OUTER UNION Operations
  - The outer union operation was developed to take the union of tuples from two relations if the relations are *not type compatible*.
  - This operation will take the 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 type compatible.
  - The attributes that are type compatible are represented only once in the result, and those attributes that are not type compatible from either relation are also kept in the result relation T(X, Y, Z).

Additional Relational Operations (continued)

- Example: An outer union can be applied to two relations whose schemas are STUDENT(Name, SSN, Department, Advisor) and INSTRUCTOR(Name, SSN, Department, Rank).
  - Tuples from the two relations are matched based on having the same combination of values of the shared attributes— Name, SSN, Department.
  - If a student is also an instructor, both Advisor and Rank will have a value; otherwise, one of these two attributes will be null.
  - The result relation STUDENT_OR_INSTRUCTOR will have the following attributes:

STUDENT_OR_INSTRUCTOR (Name, SSN, Department, Advisor, Rank)

Examples of Queries in Relational Algebra : Procedural Form

- **Retrieve the name and address of all employees who work for the 'Research' department.**

  RESEARCH_DEPT ← σ DNAME='Research' (DEPARTMENT)

  RESEARCH_EMPS ← (RESEARCH_DEPT ⋈ DNUMBER= DNOEMPLOYEE EMPLOYEE)

  RESULT ← π FNAME, LNAME, ADDRESS (RESEARCH_EMPS)

- **Retrieve the names of employees who have no dependents.**

  ALL_EMPS ← π SSN(EMPLOYEE)

  EMPS_WITH_DEPS(SSN) ← π ESSN(DEPENDENT)

  EMPS_WITHOUT_DEPS ← (ALL_EMPS - EMPS_WITH_DEPS)

  RESULT ← π LNAME, FNAME (EMPS_WITHOUT_DEPS * EMPLOYEE)

Examples of Queries in Relational Algebra – Single expressions

As a single expression, these queries become:

- **Retrieve the name and address of all employees who work for the 'Research' department.**

  $\pi_{Fname, Lname, Address}$ (σ Dname= 'Research' (DEPARTMENT $\bowtie$ Dnumber=Dno(EMPLOYEE))

- **Retrieve the names of employees who have no dependents.**

  $\pi_{Lname, Fname}((\pi_{Ssn}(EMPLOYEE) - \rho_{Ssn}(\pi\ Essn(DEPENDENT))) * EMPLOYEE)$

Types of Joins

- **Inner join**: Only returns matched records from the tables that are being joined
- **Outer join**: Matched pairs are retained and unmatched values in the other table are left null
  - **Left outer join**: Yields all of the rows in the first table, including those that do not have a matching value in the second table
  - **Right outer join**: Yields all of the rows in the second table, including those that do not have matching values in the first table

Left, Right and Full Joins

t1

a	b
a1	b1
a2	b2
a3	b3

t2

b	c
b1	c1
b2	c2
b4	c4

t1 ⟕ t2

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

t1

a	b
a1	b1
a2	b2
a3	b3

t2

b	c
b1	c1
b2	c2
b4	c4

t1 ⟖ t2

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

t1

a	b
a1	b1
a2	b2
a3	b3

t2

b	c
b1	c1
b2	c2
b4	c4

t1 ⟗ t2

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