Chapter 8:
Relational Algebra

CS-6360 Database Design

Chris Irwin Davis, Ph.D.

Email: cid021000@utdallas.edu
Phone: (972) 883-3574
Office: ECSS 4.705

Chapter 6 Outline

- Unary Relational Operations
  - SELECT (σ)
  - PROJECT (π)
- Relational Algebra Operations from Set Theory
- Binary Relational Operations
  - JOIN
  - DIVISION
- Additional Relational Operations

Chapter 6 Outline

- Examples of Queries in Relational Algebra
- Relational Calculus
  - The Tuple Relational Calculus
  - The Domain Relational Calculus

The Relational Algebra and Relational Calculus

UTD

- **Relational algebra**
  - Basic set of operations for the relational model
  - Described by E. F. Codd, IBM (1970)
- **Relational algebra expression**
  - Sequence of relational algebra operations
- **Relational calculus**
  - Higher-level declarative language for specifying relational queries
  - **Tuple Relational Calculus**
  - **Domain Relational Calculus**

Unary Relational Operations: SELECT and PROJECT

The SELECT (σ) Operation

- The SELECT Operation
  - Subset of the tuples from a relation that satisfies a selection condition:

$$\sigma_{<\text{selection condition}>}(R)$$

  - Boolean expression contains clauses of the form
    - *<attribute name> <comparison op> <constant value>*

      or

    - *<attribute name> <comparison op> <attribute name>*

The SELECT (σ) Operation

- Select the EMPLOYEE tuples whose department is 4

$$\sigma_{Dno=4}(EMPLOYEE)$$

The SELECT (σ) Operation

- Select the EMPLOYEE tuples whose department is 4

$$\sigma_{Dno=4}(EMPLOYEE)$$

- Select the EMPLOYEE tuples whose salary is greater than $30,000

$$\sigma_{Salary>30000}(EMPLOYEE)$$

The SELECT (σ) Operation

- Example:

$$\sigma_{(\text{Dno}=4 \text{ AND } \text{Salary}>25000) \text{ OR } (\text{Dno}=5 \text{ AND } \text{Salary}>30000)}(\text{EMPLOYEE})$$

- <selection condition> applied *independently* to each individual tuple *t* in *R*
  - If condition evaluates to TRUE, tuple selected
- Boolean conditions **AND**, **OR**, and **NOT**
- **Unary**
  - Applied to a single relation

The SELECT (σ) Operation

- **Selectivity**
  - The fraction of tuples selected by a selection of the condition

The SELECT (σ) Operation

- **Selectivity**
  - The fraction of tuples selected by a selection of the condition
- The **SELECT** operation commutative

$$\sigma_{<\text{cond1}>}(\sigma_{<\text{cond2}>}(R)) = \sigma_{<\text{cond2}>}(\sigma_{<\text{cond1}>}(R))$$

The SELECT (σ) Operation

- **Selectivity**
  - The fraction of **tuples** selected by a selection of the condition
- The **SELECT** operation commutative

$$\sigma_{<\text{cond1}>}(\sigma_{<\text{cond2}>}(R)) = \sigma_{<\text{cond2}>}(\sigma_{<\text{cond1}>}(R))$$

- **Cascade** SELECT operations into a single operation with **AND** condition

$$\sigma_{<\text{cond1}>}(\sigma_{<\text{cond2}>}(\ldots(\sigma_{<\text{cond}n>}(R))\ \ldots)) = \sigma_{<\text{cond1}>\ \textbf{AND}<\text{cond2}>\ \textbf{AND}\ldots\textbf{AND}\ <\text{cond}n>}(R)$$

SELECT (σ) and SQL WHERE

- In SQL, the SELECT condition is typically specified in the WHERE clause of a SQL query. For example, the following operation:

$$\sigma_{\text{Dno=4 AND Salary>25000}}\ (\text{EMPLOYEE})$$

SELECT (σ) and SQL WHERE

- In SQL, the SELECT condition is typically specified in the WHERE clause of a SQL query. For example, the following operation:

$$\sigma_{\text{Dno=4 AND Salary>25000}}\ (\text{EMPLOYEE})$$

- would correspond to the following SQL query:

```
SELECT    *
FROM      EMPLOYEE
WHERE     Dno=4 AND Salary>25000;
```

The PROJECT (π) 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, so the result of the PROJECT operation is a set of distinct tuples, and hence a valid relation. This is known as *duplicate elimination*.

The PROJECT (π) Operation

- Notice that the tuple <'F', 25000> appears only once in the following PROJECT, even though this combination of values appears twice in the EMPLOYEE relation.

$$\pi_{\text{Sex, Salary}}(\text{EMPLOYEE})$$

The PROJECT (π) Operation

- The number of tuples in a relation resulting from a PROJECT operation is always less than or equal to the number of tuples in R. If the projection list is a superkey of R—that is, it includes some key of R—the resulting relation has the same number of tuples as R. Moreover,

$$\pi_{<\text{list1}>}(\pi_{<\text{list2}>}(R)) = \pi_{<\text{list1}>}(R)$$

- as long as <list2> contains the attributes in <list1>; otherwise, the left-hand side is an incorrect expression. It is also noteworthy that commutativity does not hold on PROJECT.

PROJECT (π) and SQL SELECT

- In SQL, the PROJECT condition is typically specified in the SELECT clause of a query. For example, the following operation:

$$\pi_{\text{Sex, Salary}}(\text{EMPLOYEE})$$

PROJECT (π) and SQL SELECT

- In SQL, the PROJECT condition is typically specified in the SELECT clause of a query. For example, the following operation:

$$\pi_{\text{Sex, Salary}}(\text{EMPLOYEE})$$

- would correspond to the following SQL query:

```
SELECT    DISTINCT Sex, Salary
FROM      EMPLOYEE
```

Sequences of Operations and the RENAME Operation

Sequences of Operations and the RENAME Operation

- **In-line** expression:

$$\pi_{\text{Fname, Lname, Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))$$

Sequences of Operations and the RENAME Operation

- **In-line** expression:

$$\pi_{\text{Fname, Lname, Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))$$

- Sequence of operations:

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Salary}}(\text{DEP5\_EMPS})$$

Sequences of Operations and the RENAME Operation

- **In-line** expression:

$$\pi_{\text{Fname, Lname, Salary}}(\sigma_{\text{Dno}=5}(\text{EMPLOYEE}))$$

- Sequence of operations:

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Salary}}(\text{DEP5\_EMPS})$$

- **Rename** attributes in intermediate results
  - RENAME operation

$$\rho_{S(B1, B2, \ldots, Bn)}(R) \quad \text{or} \quad \rho_{S}(R) \quad \text{or} \quad \rho_{(B1, B2, \ldots, Bn)}(R)$$

RENAME

- **Rename** attributes in intermediate results
  - RENAME operation

$$\rho_{S(B1,\,B2,\,\ldots,\,Bn)}(R) \quad \text{or} \quad \rho_{S}(R) \quad \text{or} \quad \rho_{(B1,\,B2,\,\ldots,\,Bn)}(R)$$

- …where the symbol ρ (rho) is used to denote the RENAME operator, S is the new relation name, and B_1, B_2, ..., B_n are the new attribute names.
  - The first expression renames both the relation and its attributes,
  - The second renames the relation only, and
  - The third renames the attributes only.

Sequences of Operations and the RENAME Operation

(a)

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

Figure 6.2
Results of a sequence of operations. (a) $\pi_{Fname, Lname, Salary}(\sigma_{Dno=5}(EMPLOYEE))$. (b) Using intermediate relations and renaming of attributes.

(b)

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

R

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

Relational Algebra Operations from Set Theory

Relational Algebra Operations from Set Theory

- **UNION**, **INTERSECTION**, and **MINUS**
  - Merge the elements of two sets in various ways
  - Binary operations
  - Relations must have the same type of tuples

UNION

- $R \cup S$
- Includes all tuples that are either in R or in S or in both R and S
- Duplicate tuples eliminated

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$\text{RESULT1} \leftarrow \pi_{\text{Ssn}}(\text{DEP5\_EMPS})$$
$$\text{RESULT2(Ssn)} \leftarrow \pi_{\text{Super\_ssn}}(\text{DEP5\_EMPS})$$
$$\text{RESULT} \leftarrow \text{RESULT1} \cup \text{RESULT2}$$

UNION

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Figure 6.3
Result of the UNION operation RESULT ← RESULT1 ∪ RESULT2.

SET DIFFERENCE (or MINUS or EXCEPT)

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

INTERSECTION

- $R \cap S$
- Includes all/only tuples that are in both R and S
- Can be expressed in terms of union and difference

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

INTERSECTION and UNION

- Notice that both UNION and INTERSECTION are commutative operations; that is,

$$R \cup S = S \cup R \quad \text{and} \quad R \cap S = S \cap R$$

- Both UNION and INTERSECTION can be treated as *n*-ary operations applicable to any number of relations because both are also associative operations; that is,

$$R \cup (S \cup T) = (R \cup S) \cup T \quad \text{and} \quad (R \cap S) \cap T = R \cap (S \cap T)$$

- The MINUS operation is not commutative; that is, in general,

$$R - S \neq S - R$$

UNION, INTERSECTION, and MINUS

Figure 6.4
The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT ∪ INSTRUCTOR. (c) STUDENT ∩ INSTRUCTOR. (d) STUDENT – INSTRUCTOR. (e) INSTRUCTOR – STUDENT.

(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

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

(d)

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

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

CARTESIAN PRODUCT Operation

- **CARTESIAN PRODUCT** (CROSS PRODUCT or CROSS JOIN)
  - Denoted by ×
  - Binary set operation
  - Relations do not have to be union compatible
  - Generally not useful by itself
    - Useful when followed by a selection that matches values of attributes

Cartesian Product with σ and π

- Suppose that we want to retrieve a list of names of each female employee's dependents

$$\text{FEMALE\_EMPS} \leftarrow \sigma_{\text{Sex='F'}}(\text{EMPLOYEE})$$
$$\text{EMPNAMES} \leftarrow \pi_{\text{Fname, Lname, Ssn}}(\text{FEMALE\_EMPS})$$
$$\text{EMP\_DEPENDENTS} \leftarrow \text{EMPNAMES} \times \text{DEPENDENT}$$
$$\text{ACTUAL\_DEPENDENTS} \leftarrow \sigma_{\text{Ssn=Essn}}(\text{EMP\_DEPENDENTS})$$
$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Dependent\_name}}(\text{ACTUAL\_DEPENDENTS})$$

Binary Relational Operations: JOIN and DIVISION

Binary Operation: JOIN

- The **JOIN** Operation
  - Denoted by ⋈
  - Combine related tuples from two relations into single relation with "longer" tuples
  - *General* JOIN condition of the form <condition> **AND** <condition> **AND** … **AND** <condition>
  - Example:

$$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE}$$
$$\text{RESULT} \leftarrow \pi_{\text{Dname, Lname, Fname}}(\text{DEPT\_MGR})$$

JOIN Example

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	...

Figure 6.6
Result of the JOIN operation DEPT_MGR ← DEPARTMENT $\bowtie_{Mgr\_ssn=Ssn}$ EMPLOYEE.

JOIN

- The JOIN operation can be specified as a CARTESIAN PRODUCT operation followed by a SELECT operation. However, JOIN is very important because it is used very frequently when specifying database queries.
- Consider:

$$\text{EMP\_DEPENDENTS} \leftarrow \text{EMPNAMES} \times \text{DEPENDENT}$$
$$\text{ACTUAL\_DEPENDENTS} \leftarrow \sigma_{\text{Ssn=Essn}}(\text{EMP\_DEPENDENTS})$$

- Can be replaced with a single JOIN operation:

$$\text{ACTUAL\_DEPENDENTS} \leftarrow \text{EMPNAMES} \bowtie_{\text{Ssn=Essn}} \text{DEPENDENT}$$

JOIN General Form

- The general form of a JOIN operation on two relations $\boldsymbol{R}(A_1, A_2, ..., A_n)$ and $\boldsymbol{S}(B_1, B_2, ..., B_m)$ is

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

THETA (θ) JOIN

- **THETA JOIN**
  - Each <*condition*> of the form $A_i\ \theta\ B_j$
  - A_i is an attribute of R
  - B_j is an attribute of S
  - A_i and B_j have the same domain
  - θ (theta) is one of the comparison operators:
    - $\{=, <, \leq, >, \geq, \neq\}$

Variations of JOIN:
The EQUIJOIN and NATURAL JOIN

EQUIJOIN

- THETA JOIN where only = comparison operator used
- Always have one or more pairs of attributes that have identical values in every tuple
- Examples:
  - ssn = mgrssn
  - dno = dnumber
  - pno = pnumber

NATURAL JOIN

- Denoted by *
- Removes second (superfluous) attribute in an EQUIJOIN condition
  - e.g. “essn” in the comparison “ssn = essn”

EQUIJOIN and NATURAL JOIN

- Suppose we want to combine each PROJECT tuple with the DEPARTMENT tuple that controls the project. In the following example, first we rename the Dnumber attribute of DEPARTMENT to Dnum—so that it has the same name as the Dnum attribute in Project relation—and then we apply NATURAL JOIN:

$$PROJ\_DEPT \leftarrow PROJECT * \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$

- The same query can be done in two steps by creating an intermediate table DEPT as follows:

$$DEPT \leftarrow \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$
$$PROJ\_DEPT \leftarrow PROJECT * DEPT$$

EQUIJOIN and NATURAL JOIN

- The attribute Dnum is called the **join attribute** for the NATURAL JOIN operation
- If the attributes on which the natural join is specified *already have the same names in both relations*, renaming is unnecessary.
- For 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

EQUIJOIN and NATURAL JOIN

(a)

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

(b)

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

Figure 6.7
Results of two NATURAL JOIN operations. (a) PROJ_DEPT ← PROJECT * DEPT.
(b) DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS.

JOIN SELECTIVITY

- If no combination of tuples satisfies the JOIN condition, the result of a JOIN is an empty relation with zero tuples.
- In general, if R has n_R tuples and S has n_S tuples, the result of a JOIN operation $R \bowtie_{\text{<join condition>}} S$ will have between zero and $n_R * n_S$ tuples.
- Expected size of join result divided by the maximum size $n_R * n_S$ is a ratio called ***join selectivity***
- If there is no join condition, all combinations of tuples qualify and the JOIN degenerates into a CARTESIAN PRODUCT (also called CROSS PRODUCT or CROSS JOIN).

INNER JOIN

- A single JOIN operation is used to combine data from two relations so that related information can be presented in a single table.
- These operations are also known as **inner joins**, to distinguish them from a different join variation called outer joins (we'll get to this)
- INNER JOIN is a type of match and combine operation
- Defined formally as a combination of CARTESIAN PRODUCT and SELECTION

INNER JOIN

- The NATURAL JOIN or EQUIJOIN operation can also be specified among multiple tables, leading to an *n-way join*.
- For example, consider the following three-way join

$$((\text{PROJECT} \bowtie_{\text{Dnum=Dnumber}} \text{DEPARTMENT}) \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE})$$

- This combines each project tuple with its controlling department tuple into a single tuple, and then combines that tuple with an employee tuple that is the department manager. The net result is a consolidated relation in which each tuple contains this project-department-manager combined information.

JOIN in SQL

- In SQL, JOIN can be realized in several different ways
  - Specify the <join conditions> in the WHERE clause, along with any other selection conditions.
  - Use a nested relation
  - Explicit JOIN keyword

JOIN in SQL

- In **MySQL**, JOIN, CROSS JOIN, and INNER JOIN are syntactic equivalents (they can replace each other).
- In **standard SQL**, they are *not* equivalent. INNER JOIN is used with an ON clause, CROSS JOIN is used otherwise.

A Complete Set of Relational Algebra Operations

- Set of relational algebra operations {σ, π, ∪, ρ, –, ×} is a **complete set**
- Any relational algebra operation can be expressed as a sequence of operations from this set
  - SELECT (σ)
  - PROJECT (π)
  - UNION (∪)
  - RENAME (ρ)
  - MINUS (–) …also called "SET DIFFERENCE"
  - CARTESIAN PRODUCT (×) …also called "CROSS PRODUCT

A Complete Set of Relational Algebra Operations

- For example, the INTERSECTION operation can be expressed by using UNION and MINUS as follows

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

- Although, strictly speaking, INTERSECTION is not required, it is inconvenient to specify this complex expression every time we wish to specify an intersection.

A Complete Set of Relational Algebra Operations

- As another example, a JOIN operation can be specified as a CARTESIAN PRODUCT followed by a SELECT operation (discussed earlier):

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

A Complete Set of Relational Algebra Operations

- Similarly, a NATURAL JOIN can be specified as a CARTESIAN PRODUCT preceded by RENAME and followed by SELECT and PROJECT operations.
- Hence, the various JOIN operations are also not strictly necessary for the expressive power of the relational algebra. However, they are important to include as separate operations because they are convenient to use and are very commonly applied in database applications.
- Other operations have been included in the basic relational algebra for convenience rather than necessity.
  - One of these, the DIVISION operation…

Query Trees

Notation for Query Trees

- **Query tree**
  - Represents the input relations of query as *leaf nodes* of a tree structure
  - Represents the relational algebra operations as *internal nodes*

Query Tree

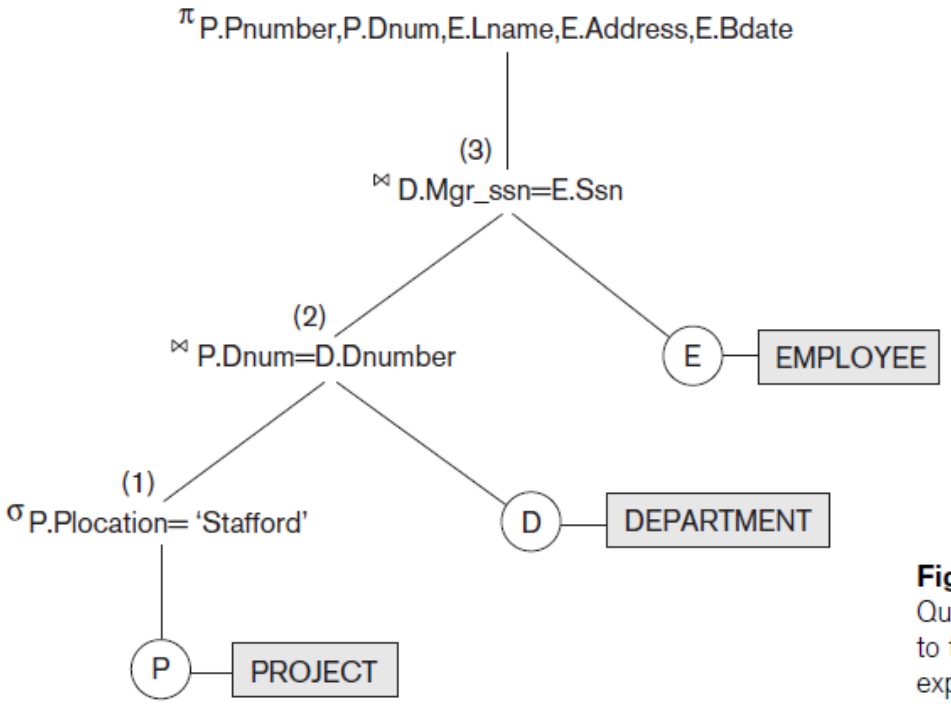

Figure 6.9
Query tree corresponding to the relational algebra expression for Q2.

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

Query Tree

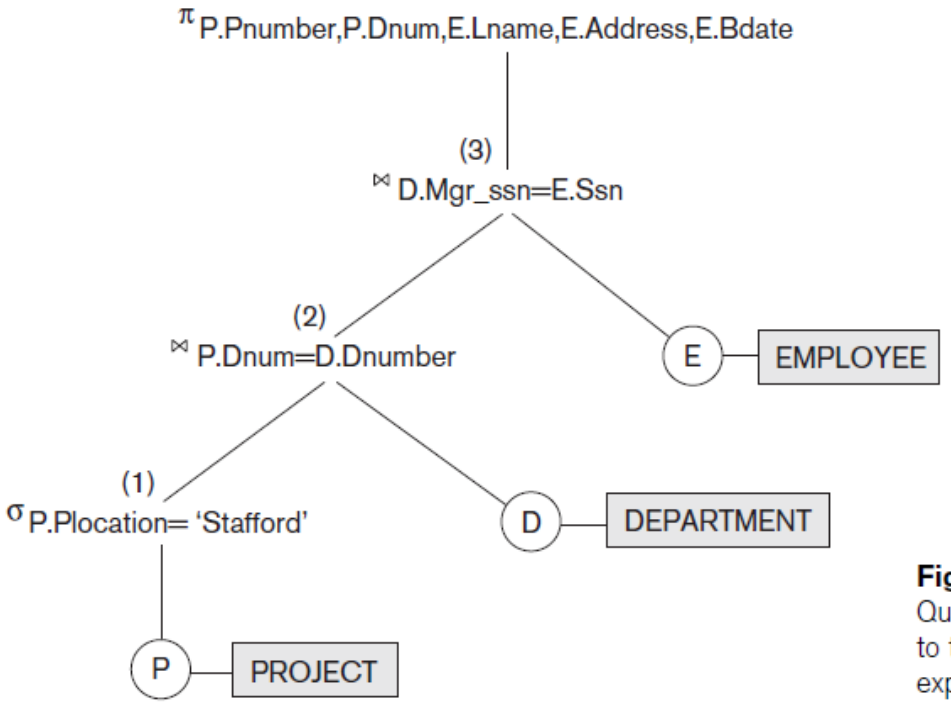

Figure 6.9
Query tree corresponding to the relational algebra expression for Q2.

```
Q2:   SELECT   Pnumber, Dnum, Lname, Address, Bdate
      FROM     PROJECT, DEPARTMENT, EMPLOYEE
      WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn AND
               Plocation='Stafford';
```

Additional Relational Operations

Generalized Projection

- Extends the projection operation by allowing functions of attributes to be included in the projection list. The generalized form can be expressed as:

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

Generalized Projection (Functions)

- 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:

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

Aggregate Functions and Grouping

- Another 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.

Aggregate Functions and Grouping

- Common functions applied to collections of numeric values, include:
  - SUM
  - AVERAGE
  - MAXIMUM
  - MINIMUM
  - COUNT

Aggregate Function

- We can define an AGGREGATE FUNCTION operation, using the symbol I (pronounced "Script F"), to specify these types of requests as follows:

$$\langle \text{grouping attributes} \rangle \; \mathfrak{I} \; \langle \text{function list} \rangle \; (R)$$

- Group tuples by the value of some of their attributes
  - Apply aggregate function independently to each group

Aggregate Function

- For example, to retrieve each department number, the number of employees in the department, and their average salary, while renaming the resulting attributes as indicated below, we write :

$$\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}(\text{Dno } \mathfrak{I} \text{ COUNT Ssn, AVERAGE Salary } (\text{EMPLOYEE}))$$

Aggregate Functions

Figure 6.10
The aggregate function operation.

a. $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}(\text{Dno } \mathfrak{I} \text{ COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE}))$.

b. $\text{Dno } \mathfrak{I} \text{ COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$.

c. $\mathfrak{I} \text{ COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$.

(a) R

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

(b)

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

(c)

Count_ssn	Average_salary
8	35125

[8]Note that this is an arbitrary notation we are suggesting. There is no standard notation.

Aggregate Function

- In the previous example, we specified a list of attribute names—between parentheses in the RENAME operation—for the resulting relation *R*.
- If no renaming is applied, then the attributes of the resulting relation that correspond to the function list will each be the concatenation of the function name with the attribute name in the form

  <*function*>_<*attribute*>.
- For example, Figure 6.10(b) (next slide) shows the result of the following operation:

$$\text{Dno}\ \mathfrak{I}\ \text{COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$$

Aggregate Functions

Figure 6.10
The aggregate function operation.

a. $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}(\text{Dno } \Im \text{ COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE}))$.

b. $\text{Dno } \Im \text{ COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$.

c. $\Im \text{ COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$.

R

(a)

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

(b)

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

(c)

Count_ssn	Average_salary
8	35125

[8]Note that this is an arbitrary notation we are suggesting. There is no standard notation.

Aggregate Function

- If no grouping attributes are specified, the functions are applied to ***all*** the tuples in the relation, so the resulting relation has a single tuple only.
- For example, Figure 6.10(c) shows the result of the following operation:

$$\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$$

Aggregate Functions

Figure 6.10
The aggregate function operation.

a. $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}({}_{\text{Dno}}\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE}))$.

b. ${}_{\text{Dno}}\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

c. $\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

R

(a)

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

(b)

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

(c)

Count_ssn	Average_salary
8	35125

[8]Note that this is an arbitrary notation we are suggesting. There is no standard notation.

Recursive Closure Operations

- Operation applied to a **recursive relationship** between tuples of *same type* (such as the relationship between an employee and a supervisor).
- This relationship is described by the foreign key Super_ssn of the EMPLOYEE relation and it relates each employee tuple (in the role of supervisee) to another employee tuple (in the role of supervisor).
  - An example of a recursive operation is to retrieve all supervisees of an employee *e* at all levels—that is, all employees e′ directly supervised by e, all employees e′ ℑ directly supervised by each employee e′, all employees e′′′ directly supervised by each employee e′′, and so on.

Recursive Closure Operations

- For example, to specify the ssns of all employees e' directly supervised—at level one—by the employee e whose name is 'James Borg', we can apply the following operation:

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

NATURAL JOIN

- MySQL implementation
- Recursive Closure

OUTER JOIN Operations

- **Outer joins**
  - Keep all tuples in *R*, or all those in *S*, or all those in both relations regardless of whether or not they have matching tuples in the other relation
  - **Types**
    - **LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN**
  - Example:

$$\text{TEMP} \leftarrow (\text{EMPLOYEE} \ ⟕_{\text{Ssn=Mgr\_ssn}} \text{DEPARTMENT})$$

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

- The following slides have equivalent queries in both SQL and Relational Algebra for comparison

Examples of Queries in Relational Algebra

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

$$\text{RESEARCH\_DEPT} \leftarrow \sigma_{\text{Dname='Research'}}(\text{DEPARTMENT})$$
$$\text{RESEARCH\_EMPS} \leftarrow (\text{RESEARCH\_DEPT} \bowtie_{\text{Dnumber=Dno}} \text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Fname, Lname, Address}}(\text{RESEARCH\_EMPS})$$

As a single in-line expression, this query becomes:

$$\pi_{\text{Fname, Lname, Address}} (\sigma_{\text{Dname='Research'}}(\text{DEPARTMENT} \bowtie_{\text{Dnumber=Dno}} (\text{EMPLOYEE}))$$

```
SELECT fname, lname, address
FROM   employee, department
WHERE  dnumber=dno AND dname='Research';
```

Examples of Queries in Relational Algebra

Query 2. 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.

$$\text{STAFFORD\_PROJS} \leftarrow \sigma_{\text{Plocation}=\text{'Stafford'}}(\text{PROJECT})$$
$$\text{CONTR\_DEPTS} \leftarrow (\text{STAFFORD\_PROJS} \bowtie_{\text{Dnum}=\text{Dnumber}} \text{DEPARTMENT})$$
$$\text{PROJ\_DEPT\_MGRS} \leftarrow (\text{CONTR\_DEPTS} \bowtie_{\text{Mgr\_ssn}=\text{Ssn}} \text{EMPLOYEE})$$
$$\text{RESULT} \leftarrow \pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(\text{PROJ\_DEPT\_MGRS})$$

```
SELECT pnumber, dnum, lname, address, bdate
FROM   project, department, employee
WHERE  dnum = dnumber AND mgrssn = ssn AND
  plocation='Stafford';
```

Examples of Queries in Relational Algebra

Query 3. Find the names of employees who work on *all* the projects controlled by department number 5.

$$\text{DEPT5\_PROJS} \leftarrow \rho_{(\text{Pno})}(\pi_{\text{Pnumber}}(\sigma_{\text{Dnum}=5}(\text{PROJECT})))$$
$$\text{EMP\_PROJ} \leftarrow \rho_{(\text{Ssn, Pno})}(\pi_{\text{Essn, Pno}}(\text{WORKS\_ON}))$$
$$\text{RESULT\_EMP\_SSNS} \leftarrow \text{EMP\_PROJ} \div \text{DEPT5\_PROJS}$$
$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{RESULT\_EMP\_SSNS} * \text{EMPLOYEE})$$

```
SELECT lname,fname
FROM   employee
WHERE  not exists (
  SELECT *
  FROM   works_on b
  WHERE  (b.pno in (SELECT pnumber FROM project WHERE
dnum=5) and not exists (
    SELECT *
    FROM   works_on c
    WHERE  ssn = c.essn and b.pno = c.pno));
```

Examples of Queries in Relational Algebra

Query 4. Make a list of project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

$SMITHS(Essn) \leftarrow \pi_{Ssn}\ (\sigma_{Lname='Smith'}(EMPLOYEE))$
$SMITH\_WORKER\_PROJS \leftarrow \pi_{Pno}(WORKS\_ON * SMITHS)$
$MGRS \leftarrow \pi_{Lname,\ Dnumber}(EMPLOYEE \bowtie_{Ssn=Mgr\_ssn} DEPARTMENT)$
$SMITH\_MANAGED\_DEPTS(Dnum) \leftarrow \pi_{Dnumber}\ (\sigma_{Lname='Smith'}(MGRS))$
$SMITH\_MGR\_PROJS(Pno) \leftarrow \pi_{Pnumber}(SMITH\_MANAGED\_DEPTS * PROJECT)$
$RESULT \leftarrow (SMITH\_WORKER\_PROJS \cup SMITH\_MGR\_PROJS)$

```
(SELECT distinct pnumber
 FROM   project,department,employee
 WHERE  dnum=dnumber and mgrssn=ssn and lname='Smith')
UNION
(SELECT pnumber
 FROM   project,works_on,employee
 WHERE  pnumber=pno and essn=ssn and lname='Smith');
```

$\pi_{Pno}\ (WORKS\_ON \bowtie_{Essn=Ssn}(\pi_{Ssn}\ (\sigma_{Lname='Smith'}(EMPLOYEE))) \cup \pi_{Pno}$
$((\pi_{Dnumber}\ (\sigma_{Lname='Smith'}(\pi_{Lname,\ Dnumber}(EMPLOYEE))) \bowtie$
$_{Ssn=Mgr\_ssn}DEPARTMENT)) \bowtie_{Dnumber=Dnum} PROJECT)$

Examples of Queries in Relational Algebra

Query 5. List the names of all employees with two or more dependents.

Strictly speaking, this query cannot be done in the *basic* (*original*) *relational algebra*. We have to use the AGGREGATE FUNCTION operation with the COUNT aggregate function. We assume that dependents of the *same* employee have *distinct* Dependent_name values.

$T1(\text{Ssn, No\_of\_dependents}) \leftarrow {}_{\text{Essn}}\Im_{\text{COUNT Dependent\_name}}(\text{DEPENDENT})$
$T2 \leftarrow \sigma_{\text{No\_of\_dependents}>2}(T1)$
$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(T2 * \text{EMPLOYEE})$

```
SELECT lname,fname
FROM   employee
WHERE  (
    SELECT COUNT(*)
    FROM dependent
    WHERE ssn=essn) >= 2;
```

Examples of Queries in Relational Algebra

Query 6. Retrieve the names of employees who have no dependents.

This is an example of the type of query that uses the MINUS (SET DIFFERENCE) operation.

$$\text{ALL\_EMPS} \leftarrow \pi_{\text{Ssn}}(\text{EMPLOYEE})$$
$$\text{EMPS\_WITH\_DEPS(Ssn)} \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT})$$
$$\text{EMPS\_WITHOUT\_DEPS} \leftarrow (\text{ALL\_EMPS} - \text{EMPS\_WITH\_DEPS})$$
$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{EMPS\_WITHOUT\_DEPS} * \text{EMPLOYEE})$$

```
SELECT fname,lname
FROM   employee
WHERE  not exists (
    SELECT *
    FROM dependent
    WHERE ssn=essn
);
```

Examples of Queries in Relational Algebra

Query 7. List the names of managers who have at least one dependent.

$$\text{MGRS}(\text{Ssn}) \leftarrow \pi_{\text{Mgr\_ssn}}(\text{DEPARTMENT})$$
$$\text{EMPS\_WITH\_DEPS}(\text{Ssn}) \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT})$$
$$\text{MGRS\_WITH\_DEPS} \leftarrow (\text{MGRS} \cap \text{EMPS\_WITH\_DEPS})$$
$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{MGRS\_WITH\_DEPS} * \text{EMPLOYEE})$$

```
SELECT fname,lname
FROM   employee
WHERE  EXISTS (
    SELECT *
    FROM department
    WHERE ssn=mgrssn)
  AND EXISTS (
    SELECT *
    FROM dependent
    WHERE ssn=essn);
```