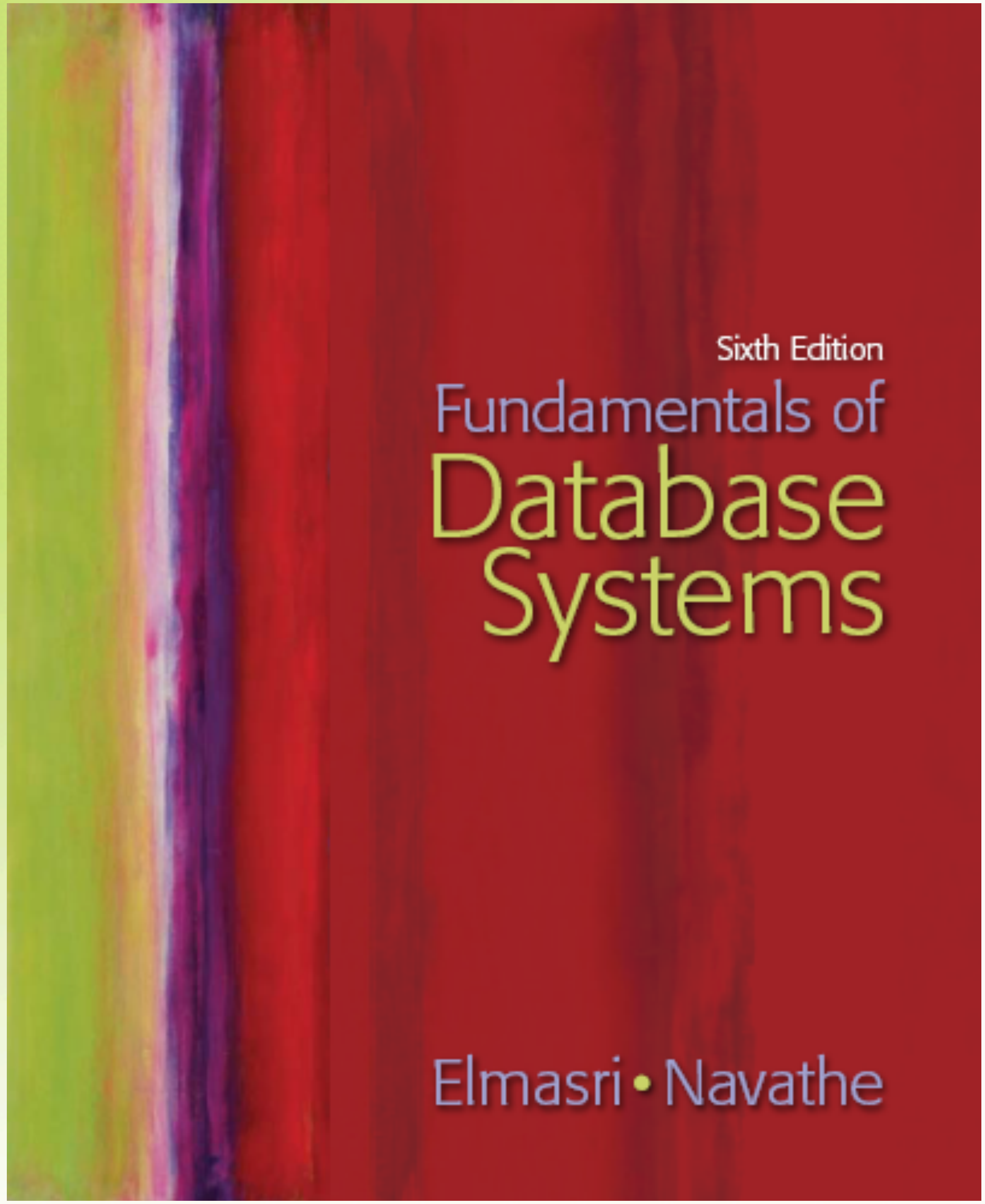
# Chapter 6

# The Relational Algebra and Relational Calculus




Addison-Wesley
is an imprint of
PEARSON

# Chapter 6 Outline

- Unary Relational Operations: SELECT and PROJECT
- Relational Algebra Operations from Set Theory
- Binary Relational Operations: JOIN and DIVISION
- Additional Relational Operations

# Chapter 6 Outline (cont'd.)

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

# The Relational Algebra and Relational Calculus

- **Relational algebra**
  - Basic set of operations for the relational model
- **Relational algebra expression**
  - Sequence of relational algebra operations
- **Relational calculus**
  - Higher-level declarative language for specifying relational queries

# Unary Relational Operations: SELECT and PROJECT

- 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>

# Unary Relational Operations: SELECT and PROJECT (cont'd.)

- Example:

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

- <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

# Unary Relational Operations: SELECT and PROJECT (cont'd.)

- **Selectivity**
  - Fraction of tuples selected by a selection condition
- SELECT operation commutative
- **Cascade** SELECT operations into a single operation with **AND** condition

# The PROJECT Operation

- Selects columns from table and discards the other columns:

$$\pi_{\langle\text{attribute list}\rangle}(R)$$

- **Degree**
  - Number of attributes in <attribute list>
- **Duplicate elimination**
  - Result of PROJECT operation is a set of distinct tuples

# 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

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

# 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

# Relational Algebra Operations from Set Theory (cont'd.)

- INTERSECTION
  - $R \cap S$
  - Includes all tuples that are in both *R* and *S*
- SET DIFFERENCE (or MINUS)
  - $R - S$
  - Includes all tuples that are in *R* but not in *S*

# The CARTESIAN PRODUCT (CROSS PRODUCT) Operation

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

# Binary Relational Operations: JOIN and DIVISION

- The **JOIN** Operation
  - Denoted by $\bowtie$
  - Combine related tuples from two relations into single "longer" tuples
  - General join condition of the form <condition> **AND** <condition> **AND**...**AND** <condition>
  - Example:

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

# Binary Relational Operations: JOIN and DIVISION (cont'd.)

- **THETA** JOIN
  - Each <condition> of the form $A_i$ θ $B_j$
  - $A_i$ 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:
    - {=, <, ≤, >, ≥, ≠}

# Variations of JOIN: The EQUIJOIN and NATURAL JOIN

- **EQUIJOIN**
  - Only = comparison operator used
  - Always have one or more pairs of attributes that have identical values in every tuple
- **NATURAL JOIN**
  - Denoted by *
  - Removes second (superfluous) attribute in an EQUIJOIN condition

# Variations of JOIN: The EQUIJOIN and NATURAL JOIN (cont'd.)

- **Join selectivity**
  - Expected size of join result divided by the maximum size $n_R * n_S$
- **Inner joins**
  - Type of match and combine operation
  - Defined formally as a combination of CARTESIAN PRODUCT and SELECTION

# 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

# The DIVISION Operation

- Denoted by ÷
- Example: retrieve the names of employees who work on all the projects that 'John Smith' works on
- Apply to relations $R(Z) \div S(X)$
  - Attributes of *R* are a subset of the attributes of *S*

# Operations of Relational Algebra

**Table 6.1** Operations of Relational Algebra

| OPERATION | PURPOSE | NOTATION |
|---|---|---|
| SELECT | Selects all tuples that satisfy the selection condition from a relation $R$. | $\sigma_{\text{<selection condition>}}(R)$ |
| PROJECT | Produces a new relation with only some of the attributes of $R$, and removes duplicate tuples. | $\pi_{\text{<attribute list>}}(R)$ |
| THETA JOIN | Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy the join condition. | $R_1 \bowtie_{\text{<join condition>}} R_2$ |
| EQUIJOIN | Produces all the combinations of tuples from $R_1$ and $R_2$ that satisfy a join condition with only equality comparisons. | $R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{(\text{<join attributes 1>}),\, (\text{<join attributes 2>})} R_2$ |
| NATURAL JOIN | Same as EQUIJOIN except that the join attributes of $R_2$ are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all. | $R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{(\text{<join attributes 1>}),\, (\text{<join attributes 2>})} R_2$ OR $R_1 * R_2$ |

# Operations of Relational Algebra (cont'd.)

**Table 6.1** Operations of Relational Algebra

| | | |
|---|---|---|
| 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 $Z = X \cup Y$. | $R_1(Z) \div R_2(Y)$ |

# Notation for Query Trees

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

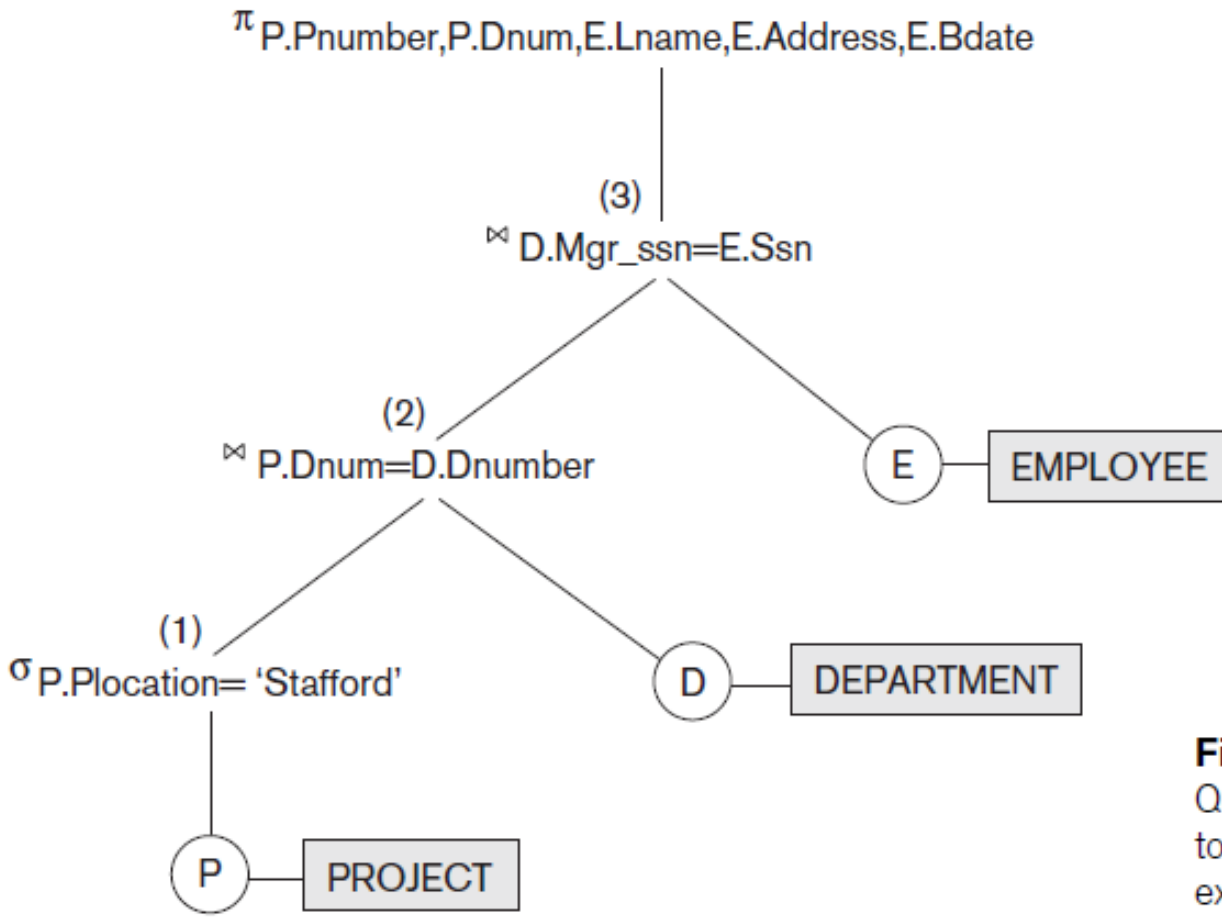



**Figure 6.9**
Query tree corresponding to the relational algebra expression for Q2.

# Additional Relational Operations

- **Generalized projection**
  - Allows functions of attributes to be included in the projection list

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

- **Aggregate functions** and **grouping**
  - Common functions applied to collections of numeric values
  - Include SUM, AVERAGE, MAXIMUM, and MINIMUM

# Additional Relational Operations (cont'd.)

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

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

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

(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.

# Recursive Closure Operations

- Operation applied to a **recursive relationship** between tuples of same type

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

# 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 OUTER UNION Operation

- Take union of tuples from two relations that have some common attributes
  - Not union (type) compatible
- **Partially compatible**
  - All tuples from both relations included in the result
  - Tut tuples with the same value combination will appear only once

# Examples of Queries in Relational Algebra

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

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

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

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

# Examples of Queries in Relational Algebra (cont'd.)

**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.

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

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

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

# Examples of Queries in Relational Algebra (cont'd.)

**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.

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

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

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

# The Tuple Relational Calculus

- Declarative expression
  - Specify a retrieval request nonprocedural language
- Any retrieval that can be specified in basic relational algebra
  - Can also be specified in relational calculus

# Tuple Variables and Range Relations

- **Tuple variables**
  - Ranges over a particular database relation
- **Satisfy** COND(*t*): $\{t \mid \text{COND}(t)\}$
- Specify:
  - **Range relation** *R* of *t*
  - Select particular combinations of tuples
  - Set of attributes to be retrieved (**requested attributes**)

# Expressions and Formulas in Tuple Relational Calculus

- General expression of tuple relational calculus is of the form:

$$\{t_1.A_j, t_2.A_k, \ldots, t_n.A_m \mid \text{COND}(t_1, t_2, \ldots, t_n, t_{n+1}, t_{n+2}, \ldots, t_{n+m})\}$$

- **Truth value** of an atom
  - Evaluates to either TRUE or FALSE for a specific combination of tuples
- **Formula** (Boolean condition)
  - Made up of one or more atoms connected via logical operators **AND**, **OR**, and **NOT**

# Existential and Universal Quantifiers

- **Universal quantifier** (∀)
- **Existential quantifier** (∃)
- Define a tuple variable in a formula as **free** or **bound**

# Sample Queries in Tuple Relational Calculus

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

**Q1:** {*t*.Fname, *t*.Lname, *t*.Address | EMPLOYEE($t$) **AND** ($\exists d$)(DEPARTMENT($d$)
**AND** *d*.Dname='Research' **AND** *d*.Dnumber=*t*.Dno)}

**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 manager of the controlling department for the project.

**Q4:** { *p*.Pnumber | PROJECT($p$) **AND** ((($\exists e$)($\exists w$)(EMPLOYEE($e$)
**AND** WORKS_ON($w$) **AND** *w*.Pno=*p*.Pnumber
**AND** e.Lname='Smith' **AND** e.Ssn=*w*.Essn) )
**OR**
(($\exists m$)($\exists d$)(EMPLOYEE($m$) **AND** DEPARTMENT($d$)
**AND** *p*.Dnum=*d*.Dnumber **AND** *d*.Mgr_ssn=*m*.Ssn
**AND** *m*.Lname='Smith')))}

# Notation for Query Graphs

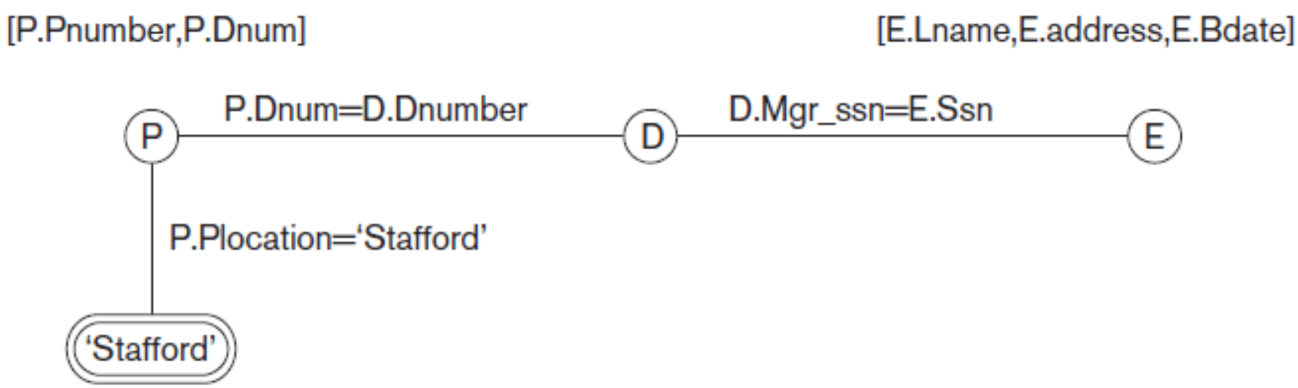



**Figure 6.13**
Query graph for Q2.

# Transforming the Universal and Existential Quantifiers

- Transform one type of quantifier into other with negation (preceded by **NOT**)
  - **AND** and **OR** replace one another
  - Negated formula becomes unnegated
  - Unnegated formula becomes negated

# Using the Universal Quantifier in Queries

**Query 3.** List the names of employees who work on *all* the projects controlled by department number 5. One way to specify this query is to use the universal quantifier as shown:

**Q3:** {$e$.Lname, $e$.Fname | EMPLOYEE($e$) **AND** (($\forall x$)(**NOT**(PROJECT($x$)) **OR NOT** ($x$.Dnum=5) **OR** (($\exists w$)(WORKS_ON($w$) **AND** $w$.Essn=$e$.Ssn **AND** $x$.Pnumber=$w$.Pno))))}

**Q3A:** {$e$.Lname, $e$.Fname | EMPLOYEE($e$) **AND** (**NOT** ($\exists x$) (PROJECT($x$) **AND** ($x$.Dnum=5) AND (**NOT** ($\exists w$)(WORKS_ON($w$) **AND** $w$.Essn=$e$.Ssn **AND** $x$.Pnumber=$w$.Pno))))}

# Safe Expressions

- Guaranteed to yield a finite number of tuples as its result
  - Otherwise expression is called **unsafe**
- Expression is **safe**
  - If all values in its result are from the domain of the expression

# The Domain Relational Calculus

- Differs from tuple calculus in type of variables used in formulas
  - Variables range over single values from domains of attributes
- Formula is made up of **atoms**
  - Evaluate to either TRUE or FALSE for a specific set of values
    - Called the **truth values** of the atoms

# The Domain Relational Calculus (cont'd.)

- QBE language
  - Based on domain relational calculus

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

**Q1:** $\{q, s, v \mid (\exists z)\ (\exists l)\ (\exists m)$ (EMPLOYEE($qrstuvwxyz$) **AND** DEPARTMENT($lmno$) **AND** $l$='Research' **AND** $m=z$)$\}$

**Query 2.** For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, birth date, and address.

**Q2:** $\{i, k, s, u, v \mid (\exists j)(\exists m)(\exists n)(\exists t)$(PROJECT($hijk$) **AND** EMPLOYEE($qrstuvwxyz$) **AND** DEPARTMENT($lmno$) **AND** $k=m$ **AND** $n=t$ **AND** $j$='Stafford')$\}$

# Summary

- Formal languages for relational model of data:
  - Relational algebra: operations, unary and binary operators
  - Some queries cannot be stated with basic relational algebra operations
    - But are important for practical use
- Relational calculus
  - Based predicate calculus