

# 204222 - Fundamentals of Database Systems

## Chapter 6

### The Relational Algebra and Relational Calculus

Adapted for 204222

by Areerat Trongratsameethong

# Chapter 6 Outline

- Unary Relational Operations: SELECT and PROJECT
- Relational Algebra Operations from Set Theory
- Binary Relational Operations: JOIN and DIVISION
- Additional Relational Operations
- Examples of Queries in Relational Algebra
- Tuple Relational Calculus
- The Domain Relational Calculus

# The Relational Algebra

- **Relational algebra**
  - Basic set of operations for the relational model
- **Relational algebra expression**
  - Sequence of relational algebra operations

# Unary Relational Operations: SELECT and PROJECT

- The SELECT Operation

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

$$\sigma_{\langle \text{selection condition} \rangle}(R)$$

$\sigma$  อ่านว่า Sigma

- 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:

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

- <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

$$\sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(R)) = \sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond1} \rangle}(R))$$

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

$$\sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(\dots(\sigma_{\langle \text{cond}n \rangle}(R)) \dots)) = \sigma_{\langle \text{cond1} \rangle \text{ AND } \langle \text{cond2} \rangle \text{ AND } \dots \text{ AND } \langle \text{cond}n \rangle}(R)$$

# The PROJECT Operation

- Selects columns from table and discards the other columns:

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

Π อ่านว่า Pi

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

Note: This was not permitted in the formal relational model, but is allowed in SQL

# Sequences of Operations and the RENAME Operation

- **In-line** expression:

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

- Sequence of operations:

DEP5\_EMPS  $\leftarrow \sigma_{Dno=5}(EMPLOYEE)$   
RESULT  $\leftarrow \pi_{Fname, Lname, Salary}(DEP5_EMPS)$

$\rho$  อ่านว่า Rho

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

$\rho_{S(B1, B2, \dots, Bn)}(R)$  or  $\rho_S(R)$  or  $\rho_{(B1, B2, \dots, Bn)}(R)$

เปลี่ยนทั้งสองอย่าง

เปลี่ยนเฉพาะชื่อ **relation**

เปลี่ยนเฉพาะชื่อ **field**



# 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  $\times$
  - 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  $\langle \text{condition} \rangle$   
**AND**  $\langle \text{condition} \rangle$  **AND...AND**  $\langle \text{condition} \rangle$
- 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  $A_i \theta B_j$
- $A_i$  is an attribute of  $R$
- $B_j$  is an attribute of  $S$
- $A_i$  and  $B_j$  have the same domain
- $\theta$  (theta) is one of the comparison operators:
  - $\{=, <, \leq, >, \geq, \neq\}$

$\theta$  อ่านว่า Theta

# 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  $\{\sigma, \pi, \cup, \rho, -, \times\}$  is a **complete set**
  - Any relational algebra operation can be expressed as a sequence of operations from this set



# The DIVISION Operation

- Denoted by  $\div$
- 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  $S$

R		S	$T \leftarrow R \div S$
A	B	A	
a1	b1	a1	
a2	b1	a2	
a3	b1	a3	
a4	b1		
a1	b2		
a3	b2		
a2	b3		
a3	b3		
a4	b3		
a1	b4		
a2	b4		
a3	b4		

T
B
b1
b4

# 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 $R$ .	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of $R$ , and removes duplicate tuples.	$\pi_{\langle \text{attribute list} \rangle}(R)$
THETA JOIN	Produces all combinations of tuples from $R_1$ and $R_2$ that satisfy the join condition.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$
EQUIJOIN	Produces all the combinations of tuples from $R_1$ and $R_2$ that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$ , OR $R_1 \bowtie_{(\langle \text{join attributes 1} \rangle, \langle \text{join attributes 2} \rangle)} R_2$
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.	$R_1 \star_{\langle \text{join condition} \rangle} R_2$ , OR $R_1 \star_{(\langle \text{join attributes 1} \rangle, \langle \text{join attributes 2} \rangle)} R_2$ OR $R_1 \star R_2$

# Operations of Relational Algebra (cont'd.)

**Table 6.1** Operations of Relational Algebra

---

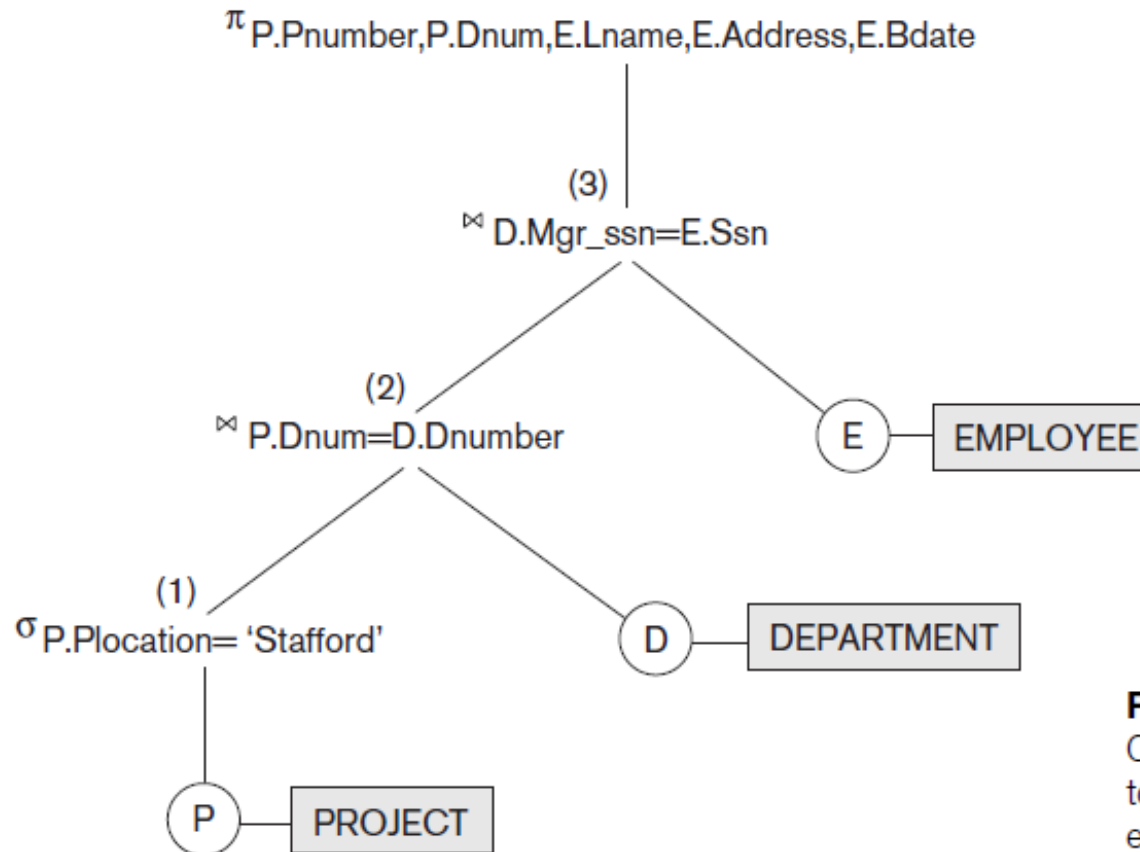
UNION	Produces a relation that includes all the tuples in $R_1$ or $R_2$ or both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both $R_1$ and $R_2$ ; $R_1$ and $R_2$ must be union compatible.	$R_1 \cap R_2$
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

$$\pi_{F_1, F_2, \dots, F_n}(R)$$

- **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

$$\langle \text{grouping attributes} \rangle \mathcal{F} \langle \text{function list} \rangle (R)$$

**Figure 6.10**

The aggregate function operation.

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

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

<sup>8</sup>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

```
BORG_SSN ← πSsn(σFname='James' AND Lname='Borg'(EMPLOYEE))  
SUPERVISION(Ssn1, Ssn2) ← πSsn, Super_ssn(EMPLOYEE)  
RESULT1(Ssn) ← πSsn1(SUPERVISION ⋈Ssn2=Ssn 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:

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

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



# 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
  - The tuples with the same value combination will appear only once

**Example:** STUDENT(Name, Ssn, Department, Advisor)  
INSTRUCTOR(Name, Ssn, Department, Rank)

STUDENT\_OR\_INSTRUCTOR(Name, Ssn, Department, Advisor, Rank)

# Examples of Queries in Relational Algebra

## EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

## DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

## DEPT\_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

## PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

## WORKS\_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

## DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

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

```
SELECT Fname, Lname, Address
FROM EMPLOYEE, DEPARTMENT
WHERE Dname = 'Research' AND Dnumber = Dno;
```

**Figure 3.5**

Schema diagram for the COMPANY relational database schema.

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

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

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

# 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.

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

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

```
DEPT5_PROJS ←  $\rho_{(Pno)}(\pi_{Pnumber}(\sigma_{Dnum=5}(PROJECT)))$   
EMP_PROJ ←  $\rho_{(Ssn, Pno)}(\pi_{Essn, Pno}(WORKS\_ON))$   
RESULT_EMP_SSNS ←  $EMP\_PROJ \div DEPT5\_PROJS$   
RESULT ←  $\pi_{Lname, Fname}(RESULT\_EMP\_SSNS * EMPLOYEE)$ 
```

ใน SQL Command ไม่มี  $R \div S$  โดยตรงต้องใช้ Operator อื่นแทน

# 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.

```
ALL_EMPS ←  $\pi_{Ssn}$ (EMPLOYEE)
EMPS_WITH_DEPS(Ssn) ←  $\pi_{Essn}$ (DEPENDENT)
EMPS_WITHOUT_DEPS ← (ALL_EMPS – EMPS_WITH_DEPS)
RESULT ←  $\pi_{Lname, Fname}$ (EMPS_WITHOUT_DEPS * EMPLOYEE)
```

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

```
MGRS(Ssn) ←  $\pi_{Mgr\_ssn}$ (DEPARTMENT)
EMPS_WITH_DEPS(Ssn) ←  $\pi_{Essn}$ (DEPENDENT)
MGRS_WITH_DEPS ← (MGRS  $\cap$  EMPS_WITH_DEPS)
RESULT ←  $\pi_{Lname, Fname}$ (MGRS_WITH_DEPS * 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**  $\text{COND}(t)$ :  $\{t \mid \text{COND}(t)\}$
- **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:

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

- **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**

**Query 0.** Retrieve the birth date and address of the employee (or employees) whose name is John B. Smith.

**Q0:**  $\{t.Bdate, t.Address \mid \text{EMPLOYEE}(t) \text{ AND } t.Fname='John' \text{ AND } t.Minit='B' \text{ AND } t.Lname='Smith'\}$

# Existential and Universal Quantifiers

- **Universal quantifier ( $\forall$ )**
- **Existential quantifier ( $\exists$ )**
- Define a tuple variable in a formula as **free** or **bound**

The ( $\exists$ ) quantifier is called an existential quantifier because a formula ( $\exists t$ )( $F$ ) is TRUE if *there exists* some tuple that makes  $F$  TRUE. For the universal quantifier,



# Sample Queries in Tuple Relational Calculus

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

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

**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.

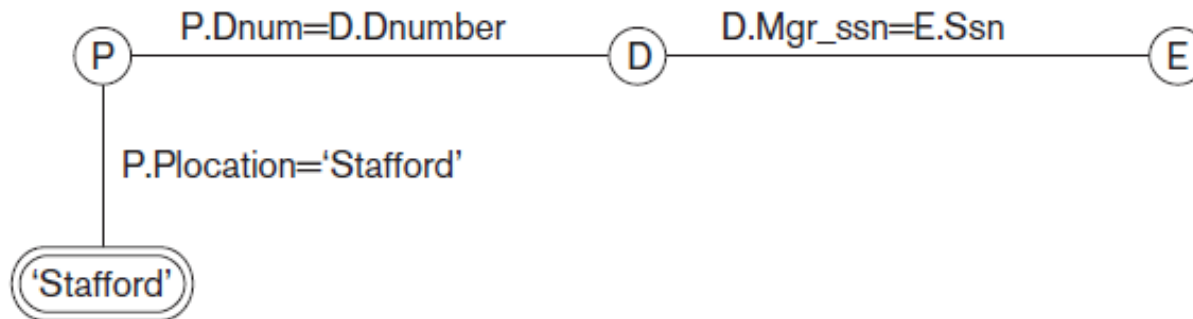
**Q4:**  $\{ p.Pnumber \mid \text{PROJECT}(p) \text{ AND } (((\exists e)(\exists w)(\text{EMPLOYEE}(e) \text{ AND } \text{WORKS\_ON}(w) \text{ AND } w.Pno=p.Pnumber \text{ AND } e.Lname='Smith' \text{ AND } e.Ssn=w.Essn) ) \text{ OR } ((\exists m)(\exists d)(\text{EMPLOYEE}(m) \text{ AND } \text{DEPARTMENT}(d) \text{ AND } p.Dnum=d.Dnumber \text{ AND } d.Mgr\_ssn=m.Ssn \text{ AND } m.Lname='Smith'))))\}$

# Notation for Query Graphs

[P.Pnumber,P.Dnum]

[E.Lname,E.address,E.Bdate]

**Figure 6.13**  
Query graph for Q2.



**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.

# 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

$$(\forall x) (P(x)) \equiv \text{NOT } (\exists x) (\text{NOT } (P(x)))$$

$$(\exists x) (P(x)) \equiv \text{NOT } (\forall x) (\text{NOT } (P(x)))$$

$$(\forall x) (P(x) \text{ AND } Q(x)) \equiv \text{NOT } (\exists x) (\text{NOT } (P(x)) \text{ OR } \text{NOT } (Q(x)))$$

$$(\forall x) (P(x) \text{ OR } Q(x)) \equiv \text{NOT } (\exists x) (\text{NOT } (P(x)) \text{ AND } \text{NOT } (Q(x)))$$

$$(\exists x) (P(x) \text{ OR } Q(x)) \equiv \text{NOT } (\forall x) (\text{NOT } (P(x)) \text{ AND } \text{NOT } (Q(x)))$$

$$(\exists x) (P(x) \text{ AND } Q(x)) \equiv \text{NOT } (\forall x) (\text{NOT } (P(x)) \text{ OR } \text{NOT } (Q(x)))$$

Notice also that the following is TRUE, where the  $\Rightarrow$  symbol stands for **implies**:

$$(\forall x)(P(x)) \Rightarrow (\exists x)(P(x))$$

$$\text{NOT } (\exists x)(P(x)) \Rightarrow \text{NOT } (\forall x)(P(x))$$

# 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:**  $\{e.Lname, e.Fname \mid \text{EMPLOYEE}(e) \text{ AND } ((\forall x)(\text{NOT}(\text{PROJECT}(x)) \text{ OR NOT } (x.Dnum=5) \text{ OR } ((\exists w)(\text{WORKS\_ON}(w) \text{ AND } w.Essn=e.Ssn \text{ AND } x.Pnumber=w.Pno))))))\}$

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

**Query 3'.** List the name of each employee who works on *some* project controlled by department number 5. This is a variation of Q3 in which *all* is changed to *some*. In this case we need two join conditions and two existential quantifiers.

**Q0':**  $\{e.Lname, e.Fname \mid \text{EMPLOYEE}(e) \text{ AND } ((\exists x)(\exists w)(\text{PROJECT}(x) \text{ AND } \text{WORKS\_ON}(w) \text{ AND } x.Dnum=5 \text{ AND } w.Essn=e.Ssn \text{ AND } x.Pnumber=w.Pno)))\}$

# 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

For example, the expression

$\{t \mid \text{NOT}(\text{EMPLOYEE}(t))\}$

is *unsafe* because it yields all tuples in the universe that are *not* EMPLOYEE tuples

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

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

# 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