Fundamentals of Database Systems **Chapter8:**

**The Relational Algebra**

****

Chapter Outline

◼ Relational Algebra

◼ Unary Relational Operations

◼ Relational Algebra Operations From Set Theory ◼ Binary Relational Operations

◼ Additional Relational Operations

◼ Examples of Queries in Relational Algebra ◼ Relational Calculus

◼ Tuple Relational Calculus

◼ Domain Relational Calculus

◼ Example Database Application (COMPANY)

**Slide 8- 2**

Relational Algebra Overview

◼ Relational algebra is the basic set of operations for the relational model

◼ These operations enable a user to specify **basic retrieval requests** (or **queries**)

◼ The result of an operation is a *new relation*, which may have been formed from one or more *input* relations

◼ This property makes the algebra “closed” (all objects in relational algebra are relations)

**Slide 8- 3**

Relational Algebra Overview (continued)

◼ The **algebra operations** thus produce new relations

◼ These can be further manipulated using operations of the same algebra

◼ A sequence of relational algebra operations forms a **relational algebra expression** ◼ The result of a relational algebra expression is also a relation that represents the result of a database query (or retrieval request)

**Slide 8- 4**

Relational Algebra Overview

◼ Relational Algebra consists of several groups of operations ◼ Unary Relational Operations

◼ SELECT (symbol: σ (sigma))

◼ PROJECT (symbol: π (pi))

◼ RENAME (symbol: ρ (rho))

◼ Relational Algebra Operations From Set Theory

◼ UNION ( ∪ ), INTERSECTION ( ∩ ), DIFFERENCE (or MINUS, **–** ) ◼ CARTESIAN PRODUCT ( **x** )

◼ Binary Relational Operations

◼ JOIN (several variations of JOIN exist)

◼ DIVISION

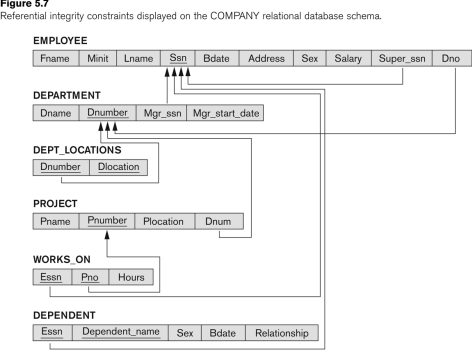
◼ Additional Relational Operations

◼ OUTER JOINS, OUTER UNION

◼ AGGREGATE FUNCTIONS (These compute summary of information: for example, SUM, COUNT, AVG, MIN, MAX)

**Slide 8- 5**

Database State for COMPANY

◼ All examples discussed below refer to the COMPANY database shown here.

**Slide 8- 6**

Unary Relational Operations: SELECT

◼ The SELECT operation (denoted by σ (sigma)) is used to select a *subset* of the tuples from a relation based on a **selection condition**. ◼ The selection condition acts as a **filter**

◼ Keeps only those tuples that satisfy the qualifying condition ◼ Tuples satisfying the condition are *selected* whereas the other tuples are discarded (*filtered out*)

◼ Examples:

◼ Select the EMPLOYEE tuples whose department number is 4: σ DNO = 4 (EMPLOYEE)

◼ Select the employee tuples whose salary is greater than $30,000: σ SALARY > 30,000 (EMPLOYEE)

**Slide 8- 7**

Unary Relational Operations: SELECT

◼ In general, the *select* operation is denoted by σ <selection condition>(R) where

◼ the symbol σ (sigma) is used to denote the *select* operator

◼ the selection condition is a Boolean (conditional) expression specified on the attributes of relation R ◼ tuples that make the condition **true** are selected ◼ appear in the result of the operation

◼ tuples that make the condition **false** are filtered out ◼ discarded from the result of the operation

**Slide 8- 8**

Unary Relational Operations: SELECT (continued)

◼ SELECT Operation Properties

◼ The SELECT operation σ <selection condition>(R) produces a relation S that has the same schema (same attributes) as R

◼ SELECT σ is commutative:

◼ σ <condition1>(σ < condition2> (R)) = σ <condition2> (σ < condition1> (R)) ◼ Because of commutativity property, a cascade (sequence) of SELECT operations may be applied in any order:

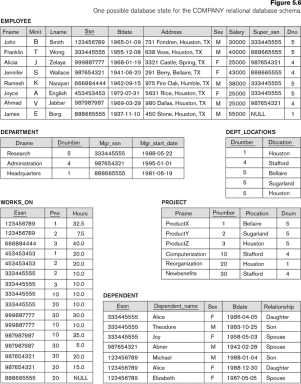
◼ σ<cond1>(σ<cond2> (σ<cond3> (R)) = σ<cond2> (σ<cond3> (σ<cond1> ( R))) ◼ A cascade of SELECT operations may be replaced by a single selection with a conjunction of all the conditions:

◼ σ<cond1>(σ< cond2> (σ<cond3>(R)) = σ <cond1> AND < cond2> AND <  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

**Slide 8- 9**

The following query results refer to this database state



**Slide 8- 10**

Unary Relational Operations: PROJECT

◼ PROJECT Operation is denoted by π (pi) ◼ This operation keeps certain *columns* (attributes) from a relation and discards the other columns. ◼ PROJECT creates a vertical partitioning ◼ The list of specified columns (attributes) is kept in each tuple

◼ The other attributes in each tuple are discarded ◼ Example: To list each employee’s first and last name and salary, the following is used: πLNAME, FNAME,SALARY(EMPLOYEE)

**Slide 8- 11**

Unary Relational Operations: PROJECT (cont.)

◼ The general form of the *project* operation is: π<attribute list>(R)

◼ π (pi) is the symbol used to represent the *project* operation

◼ <attribute list> is the desired list of attributes from relation R.

◼ The project operation *removes any duplicate tuples*

◼ This is because the result of the *project* operation must be a *set of tuples*

◼ Mathematical sets *do not allow* duplicate elements.

**Slide 8- 12**

Unary Relational Operations: PROJECT (contd.)

◼ PROJECT Operation Properties

◼ The number of tuples in the result of projection π<list>(R) is always less or equal to the number of tuples in R

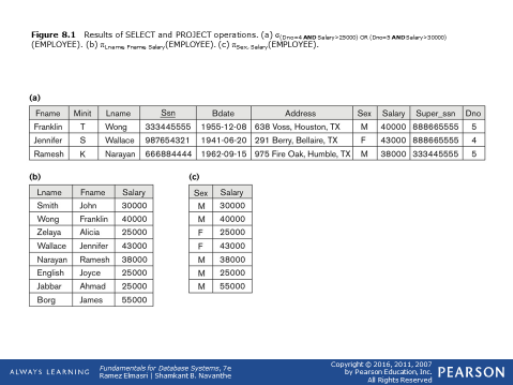
◼ If the list of attributes includes a *key* of R, then the number of tuples in the result of PROJECT is *equal* to the number of tuples in R

◼ PROJECT is *not* commutative

◼ π <list1> (π <list2> (R) ) = π <list1> (R) as long as <list2> contains the attributes in <list1>

**Slide 8- 13**

Examples of applying SELECT and PROJECT operations

**Slide 8- 14**

Relational Algebra Expressions

◼ We may want to apply several relational algebra operations one after the other

◼ Either we can write the operations as a single **relational algebra expression** by nesting the operations, or

◼ We can apply one operation at a time and create **intermediate result relations**.

◼ In the latter case, we must give names to the relations that hold the intermediate results.

**Slide 8- 15**

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:

◼ πFNAME, LNAME, SALARY(σ DNO=5(EMPLOYEE))

◼ OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:

◼ DEP5\_EMPS ← σ DNO=5(EMPLOYEE)

◼ RESULT ← π FNAME, LNAME, SALARY (DEP5\_EMPS)

**Slide 8- 16**

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 (see JOIN operation later)

**Slide 8- 17**

Unary Relational Operations: RENAME (continued)

◼ The general RENAME operation ρ can be expressed by any of the following forms: ◼ ρS (B1, B2, …, Bn )(R) changes both:

◼ the relation name to S, *and*

◼ the column (attribute) names to B1, B1, …..Bn ◼ ρS(R) changes:

◼ the *relation name* only to S

◼ ρ(B1, B2, …, Bn )(R) changes:

◼ the *column (attribute) names* only to B1, B1, …..Bn

**Slide 8- 18**

Unary Relational Operations: RENAME (continued)

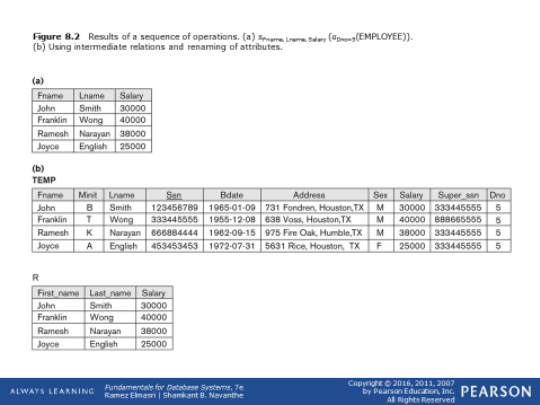
◼ For convenience, we also use a *shorthand* for renaming attributes in an intermediate relation: ◼ If we write:

• RESULT ← π 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)← ρ RESULT (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 ← symbol is an assignment operator

**Slide 8- 19**

Example of applying multiple operations and RENAME



**Slide 8- 20**

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)

**Slide 8- 21**

Relational Algebra Operations from Set Theory: UNION

◼ Example:

◼ To retrieve the social security numbers of all employees who either *work in department 5* (RESULT1 below) or *directly supervise an employee who works in department 5* (RESULT2 below)

◼ We can use the UNION operation as follows:

DEP5\_EMPS ← σDNO=5 (EMPLOYEE)

RESULT1 ← π SSN(DEP5\_EMPS)

RESULT2(SSN) ← πSUPERSSN(DEP5\_EMPS)

RESULT ← RESULT1 ∪ RESULT2

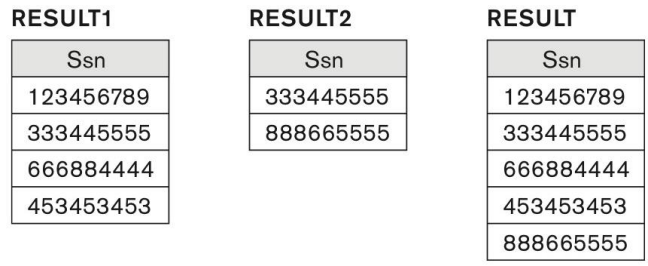
◼ The union operation produces the tuples that are in either RESULT1 or RESULT2 or both

**Slide 8- 22**

**Figure 8.3** Result of the UNION

operation RESULT ← RESULT1 ∪

RESULT2.

**Slide 8- 25**

Relational Algebra Operations from Set Theory

◼ Type Compatibility of operands is required for the binary set operation UNION ∪, (also for INTERSECTION ∩, and SET DIFFERENCE –, see next slides)

◼ 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∪R2 (also for R1∩R2, or R1– R2, see next slides) has the same attribute names as the *first* operand relation R1 (by convention)

**Slide 8- 24**

Relational Algebra Operations from Set Theory: INTERSECTION

◼ INTERSECTION is denoted by ∩ ◼ The result of the operation R ∩ 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”

**Slide 8- 25**

Relational Algebra Operations from Set Theory: SET DIFFERENCE (cont.)

◼ SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by –

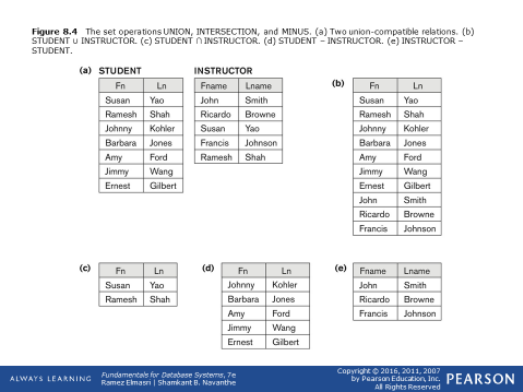
◼ The result of R – S, is a relation that includes all tuples that are in R but not in S

◼ 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”

**Slide 8- 26**

Example to illustrate the result of UNION, INTERSECT, and DIFFERENCE

**Slide 8- 27**

Some properties of UNION, INTERSECT, and DIFFERENCE

◼ Notice that both union and intersection are *commutative* operations; that is

◼ R ∪ S = S ∪ R, and R ∩ S = S ∩ 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 ∪ (S ∪ T) = (R ∪ S) ∪ T

◼ (R ∩ S) ∩ T = R ∩ (S ∩ T)

◼ The minus operation is not commutative; that is, in general

◼ R – S ≠ S – R

**Slide 8- 28**

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 nRtuples (denoted as |R| = nR), and S has nStuples, then R x S will have nR\* nStuples.

◼ The two operands do NOT have to be "type compatible”

**Slide 8- 29**

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT (cont.)

◼ Generally, CROSS PRODUCT is not a meaningful operation

◼ Can become meaningful when followed by other operations

◼ Example (not meaningful):

◼ FEMALE\_EMPS ← σ SEX=’F’(EMPLOYEE)

◼ EMPNAMES ← π FNAME, LNAME, SSN (FEMALE\_EMPS) ◼ EMP\_DEPENDENTS ← EMPNAMES x DEPENDENT ◼ EMP\_DEPENDENTS will contain every combination of EMPNAMES and DEPENDENT

◼ whether or not they are actually related

**Slide 8- 30**

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT (cont.)

◼ To keep only combinations where the DEPENDENT is related to the EMPLOYEE, we add a SELECT operation as follows

◼ Example (meaningful):

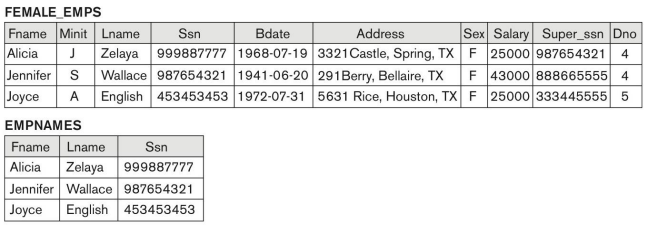
◼ FEMALE\_EMPS ← σ SEX=’F’(EMPLOYEE)

◼ EMPNAMES ← π FNAME, LNAME, SSN (FEMALE\_EMPS) ◼ EMP\_DEPENDENTS ← EMPNAMES x DEPENDENT ◼ ACTUAL\_DEPS ← σ SSN=ESSN(EMP\_DEPENDENTS) ◼ RESULT ← π FNAME, LNAME, DEPENDENT\_NAME (ACTUAL\_DEPS)

◼ RESULT will now contain the name of female employees and their dependents

**Slide 8- 31**

**Figure 8.5** The CARTESIAN PRODUCT (CROSS PRODUCT) operation.



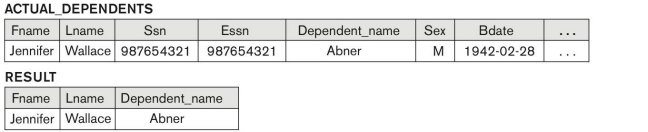
*continued on next slide*

**Slide 8- 32**

**Figure 8.5 (continued)** The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

*continued on next slide* **Slide 8- 33**

**Figure 8.5 (continued)** The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

**Slide 8- 34**

Binary Relational Operations: JOIN

◼ JOIN Operation (denoted by )

◼ 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 <join condition>S

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

**Slide 8- 35**

Binary Relational Operations: JOIN (cont.)

◼ 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. ◼ We do this by using the join operation.

◼ DEPT\_MGR ← DEPARTMENT MGRSSN=SSN EMPLOYEE ◼ MGRSSN=SSN is the join condition

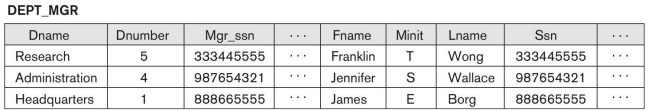
◼ Combines each department record with the employee who manages the department

◼ The join condition can also be specified as

DEPARTMENT.MGRSSN= EMPLOYEE.SSN

**Slide 8- 36**

**Figure 8.6** Result of the JOIN operation DEPT\_MGR ← DEPARTMENT|X| Mgr\_ssn=SsnEMPLOYEE.

**Slide 8- 37**

Some properties of JOIN

◼ Consider the following JOIN operation:

◼ R(A1, A2, . . ., An) S(B1, B2, . . ., Bm) R.Ai=S.Bj

◼ 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 nRtuples, and S has nStuples, then the join result will generally have *less than* nR\* nStuples.

◼ Only related tuples (based on the join condition) will appear in the result

**Slide 8- 38**

Some properties of JOIN

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

*theta*

◼ 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

**Slide 8- 39**

Binary Relational Operations: EQUIJOIN

◼ EQUIJOIN Operation

◼ 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.

◼ The JOIN seen in the previous example was an EQUIJOIN.

**Slide 8- 40**

Binary Relational Operations:

NATURAL JOIN Operation

◼ NATURAL JOIN Operation

◼ 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.

**Slide 8- 41**

Binary Relational Operations

NATURAL JOIN (continued)

◼ 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

◼ Another example: Q ← R(A,B,C,D) \* S(C,D,E)

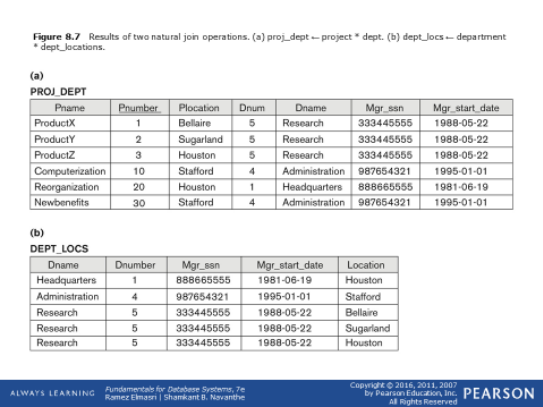
◼ The implicit join condition includes *each pair* of attributes with the same name, “AND”ed together:

◼ R.C=S.C AND R.D.S.D

◼ Result keeps only one attribute of each such pair:

◼ Q(A,B,C,D,E)

**Slide 8- 42**

Example of NATURAL JOIN operation

**Slide 8- 43**

Complete Set of Relational Operations

◼ The set of operations including SELECT σ, PROJECT π , UNION ∪, 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 ∩ S = (R ∪ S ) – ((R − S) ∪ (S − R))

◼ R <join condition>S = σ <join condition> (R X S)

**Slide 8- 44**

Binary Relational Operations: DIVISION

◼ DIVISION Operation

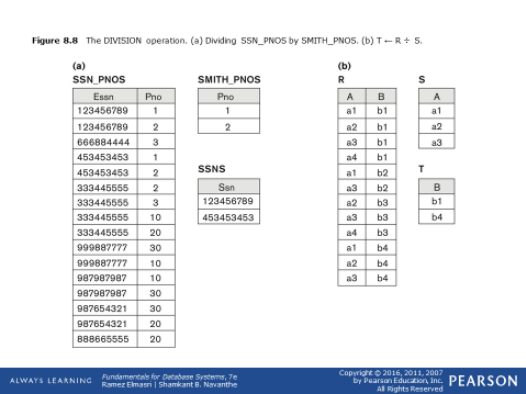
◼ The division operation is applied to two relations ◼ R(Z) ÷ S(X), where X subset Z. Let Y = Z - X (and hence Z = X ∪ Y); that is, let Y be the set of attributes of R that are not attributes of S.

◼ The result of DIVISION is a relation T(Y) that includes a tuple t if tuples tR appear in R with tR[Y] = t, and with ◼ tR[X] = ts*for every tuple* tsin 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.

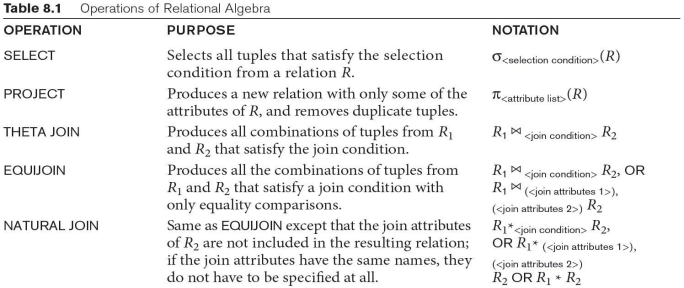
**Slide 8- 45**

Example of DIVISION



**Slide 8- 46**

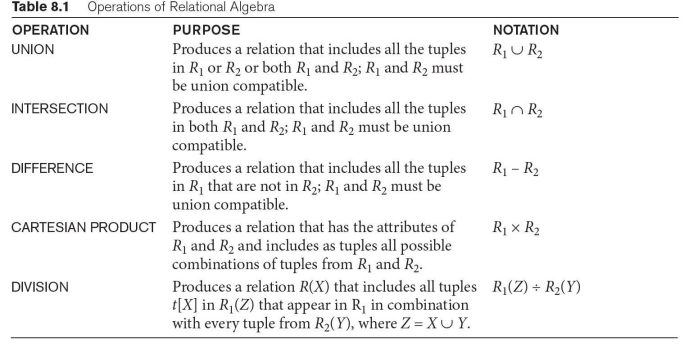
**Table 8.1** Operations of Relational Algebra



*continued on next slide*

**Slide 8- 47**

**Table 8.1** Operations of Relational Algebra (continued)

**Slide 8- 48**