# Chapter 8
# The Relational Algebra

# Content

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

# Introduction

- **relational algebra:**
  - Defines the basic set of operations for a relational model
  - Theoretical model designed via using mathematical expressions to perform a set of operations on a relational
  - These operations are algebraic operations
  - Also termed as Procedural query language
  - ADV:
    - Provide formal foundation of RDBMS
    - Used as a basis for implementing & processing query optimization module
    - to write optimized queries by understanding DB operations in more details
- **relational algebra expressions**
  - A sequence of relational algebra operations
  - Input & output both are relations

- **relational algebra operations:**
  - **Fundamental operations:**
    - Select $\delta$
    - Project $\pi$
    - Union $\cup$
    - Set difference -
    - Cartesian product/cross product X
    - Rename $\rho$
  - **Additional operations:**
    - Set intersection $\cap$
    - Assignment $\leftarrow$
    - Join $\bowtie$
    - Division /

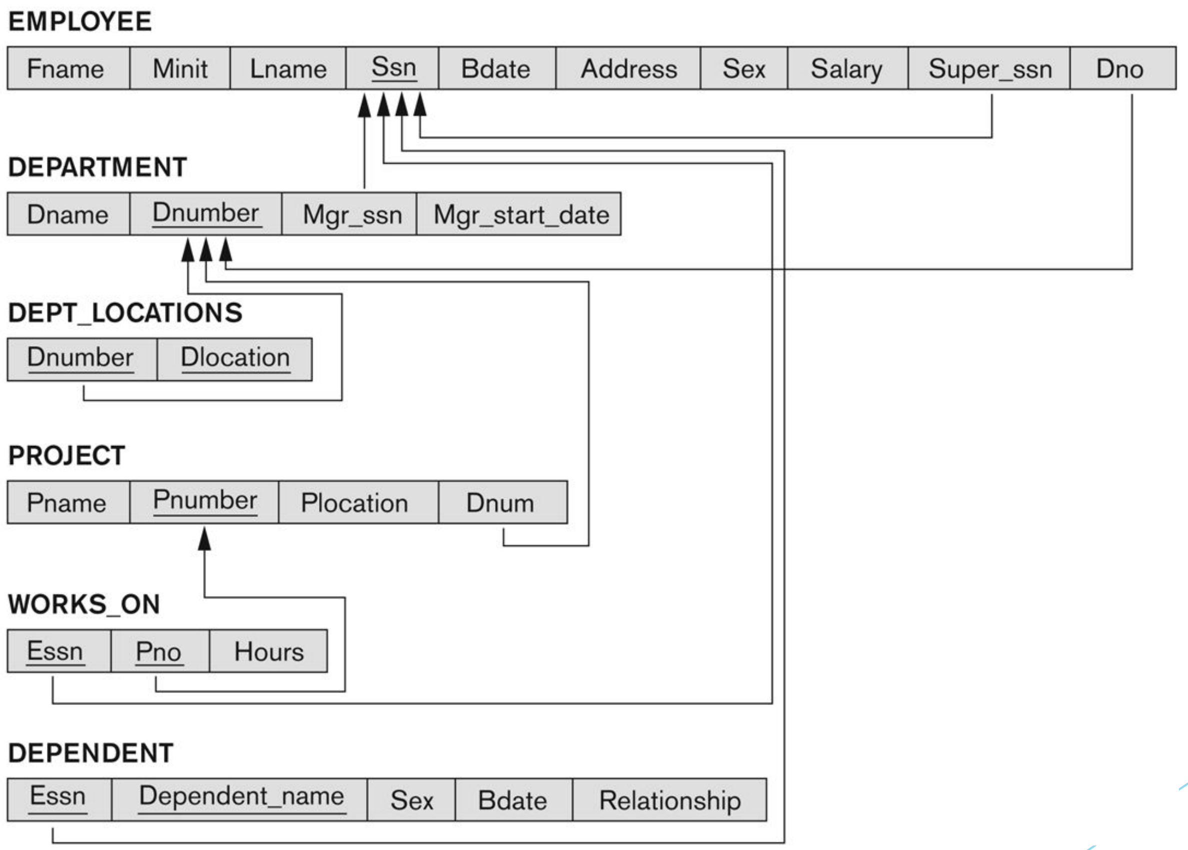
# Relational Algebra Operations

- Operations can be divided into two groups.
- Since a relation is a set of tuples, so One group includes set operations: Set operations include **UNION, INTERSECTION, SET DIFFERENCE, and CARTESIAN PRODUCT (also known as CROSS PRODUCT).**
- The other group consists of operations developed specifically for relational database include **SELECT, PROJECT, and JOIN**.
- SELECT and PROJECT are **unary operations** that operate on single relations.
- JOIN operates on **two tables** by combining related tuples (records) based on join conditions.

# Database State for COMPANY

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

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



# Unary Relational Operations: SELECT and PROJECT

- ***<u>The SELECT Operation</u>***
- The SELECT operation is used to choose a **subset of the tuples from a relation that satisfies a selection condition**.
- Alternatively, we can consider the SELECT operation to restrict the tuples in a relation to only those tuples that satisfy the condition.
- The SELECT operation is denoted by:

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

- where the symbol σ (sigma) is used to denote the SELECT operator and the Selection condition is a Boolean expression (condition) specified on the attributes of relation R.
- The relation resulting from the SELECT operation has the same attributes as R.

# Unary Relational Operations: SELECT and PROJECT

- ***The SELECT Operation***
- For example:
- to select the EMPLOYEE tuples whose department is 4,

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

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

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

# Unary Relational Operations: SELECT and PROJECT

- ***The SELECT Operation***
- The Boolean expression specified in <selection condition> is made up of a number of clauses of the form
  - <attribute name> <comparison op> <constant value> e.g., Dno>4
- or
  - <attribute name> <comparison op> <attribute name> e.g., hiredate > dateofBirth
- where <attribute name> is the name of an attribute of R,
- <comparison op> is normally one of the operators {=, <, ≤, >, ≥, ≠},
- and <constant value> is a constant value from the attribute domain.
- Clauses can be connected by the standard Boolean operators and, or, and not to form a general selection condition.

# Unary Relational Operations: SELECT and PROJECT

- ***The SELECT Operation***
- For example, to select the tuples for all employees who
  - either work in department 4 and make over \$25,000 per year,
  - or work in department 5 and make over \$30,000,
- we can specify the following SELECT operation:

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

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno |
|---|---|---|---|---|---|---|---|---|---|
| Franklin | T | Wong | 333445555 | 1955-12-08 | 638 Voss, Houston, TX | M | 40000 | 888665555 | 5 |
| Jennifer | S | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX | F | 43000 | 888665555 | 4 |
| Ramesh | K | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M | 38000 | 333445555 | 5 |

# Unary Relational Operations: SELECT and PROJECT

- ***The SELECT Operation***
- Comparison operators used between attributes when domains are ordered values:
  - {=, <, ≤, >, ≥, ≠}
  - E.g., of ordered value domains are numeric or float data
- Comparison operators used between attributes when domains are unordered values:
  - {=, ≠}
  - E.g., of an unordered domain is Color = { ‘red’, ‘blue’, ‘green’, ‘white’, ‘yellow’, …},
  - where **no order is specified among the various colors**.

# Unary Relational Operations: SELECT and PROJECT

- ***The SELECT Operation***
- The <selection condition> is applied independently to each individual tuple t in R.
- If the condition evaluates to TRUE, then tuple t is selected.
- All the selected tuples appear in the result of the SELECT operation.
- The Boolean conditions AND, OR, and NOT have their normal interpretation, as follows:

(cond1 **AND** cond2) is TRUE if both (cond1) and (cond2) are TRUE; otherwise, it is FALSE.

(cond1 **OR** cond2) is TRUE if either (cond1) or (cond2) or both are TRUE; otherwise, it is FALSE.

(**NOT** cond) is TRUE if cond is FALSE; otherwise, it is FALSE.

# Unary Relational Operations: SELECT and PROJECT

- ***The SELECT Operation***
- The **SELECT operator is unary**;
  - that is, it is applied to a single relation.
- The **degree of the relation** resulting from a SELECT operation—its number of attributes—is the same as the degree of R.

# Unary Relational Operations: SELECT and PROJECT

- ***The SELECT Operation***
- **SELECT operation is commutative;** that is,
  - $\sigma_{<cond1>}(\sigma_{<cond2>}(R)) = \sigma_{<cond2>}(\sigma_{<cond1>}(R))$
- Hence, **a sequence of SELECTs can be applied in any order**.
- In addition, **we can always combine a sequence of SELECT operations into a single SELECT operation with a conjunctive (AND) condition**; that is,
- $\sigma_{<cond1>}(\sigma_{<cond2>}(\ldots (\sigma_{<condn>}(R)) \ldots)) = \sigma_{<cond1> \text{ AND } <cond2> \text{ AND} \ldots \text{AND } <condn>}(R)$

# Unary Relational Operations: SELECT and PROJECT

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

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

- would correspond to the following SQL query:

```
SELECT *
FROM EMPLOYEE
WHERE Dno=4;
```

# Unary Relational Operations: SELECT and PROJECT

- *The PROJECT Operation*
- **SELECT operation** chooses some of the rows from the table while discarding other rows.
- **PROJECT operation** selects certain columns from the table and discards the other columns.
- If we are **interested in only certain attributes of a relation, we use the PROJECT** operation to project the relation over these attributes only.
- The general form of the PROJECT operation is:

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

- where π (pi) is the symbol used to represent the PROJECT operation, and <attribute list> is the desired list of attributes from the attributes of relation R.
- **result of the PROJECT** operation
  - Relation with specified attributes in the listed order
  - its degree of relation = the number of attributes in <attribute list>.

# Unary Relational Operations: SELECT and PROJECT

- ***The PROJECT Operation***
- If we are **interested in only certain attributes of a relation, we use the PROJECT** operation to project the relation over these attributes only.
- List each employee’s first and last name and salary, we can use the PROJECT operation as follows

$$\pi_{\text{Lname, Fname, Salary}}(\text{EMPLOYEE})$$

| Lname | Fname | Salary |
|---|---|---|
| Smith | John | 30000 |
| Wong | Franklin | 40000 |
| Zelaya | Alicia | 25000 |
| Wallace | Jennifer | 43000 |
| Narayan | Ramesh | 38000 |
| English | Joyce | 25000 |
| Jabbar | Ahmad | 25000 |
| Borg | James | 55000 |

# Unary Relational Operations: SELECT and PROJECT

- ***The PROJECT Operation***
- The PROJECT operation **removes any duplicate 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**.

$$\pi_{age,\ salary}(EMPLOYEE)$$

# Unary Relational Operations: SELECT and PROJECT

- ***The PROJECT Operation***
- **The number of tuples in a relation resulting from a PROJECT operation is always <= the number of tuples in R.**
  - because duplicates are eliminated.
- In SQL, the PROJECT attribute list is specified in the SELECT clause of a query.
- For example, the following operation:
  - $\pi_{age, salary}(EMPLOYEE)$
  - Will be represented in SQL as :
  - SELECT DISTINCT AGE,SALARY FROM EMPLOYEE

# Unary Relational Operations: SELECT and PROJECT

- ***Sequences of Operations and the RENAME Operation***
- 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 this case , we must give names to the relations that hold the intermediate results.

# Unary Relational Operations: SELECT and PROJECT

- ***Sequences of Operations and the RENAME Operation***
- **For 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.**
- **Option 1**: We can write a single relational algebra expression, also known as an in-line expression, as follows:

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

# Unary Relational Operations: SELECT and PROJECT

- ***Sequences of Operations and the RENAME Operation***
- **For 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.**
- **Option 2**: We can show the sequence of operations, giving a name to each intermediate relation, and using the assignment operation

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

- We can also use this technique to **rename the attributes** in the intermediate and result relations.

# Unary Relational Operations: SELECT and PROJECT

- ***Sequences of Operations and the RENAME Operation***
- **To rename the attributes in a relation,**
  - **list the new attribute names in parentheses**, as:

$$\text{TEMP} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$$
$$R(\text{First\_name, Last\_name, Salary}) \leftarrow \pi_{\text{Fname, Lname, Salary}}(\text{TEMP})$$

**TEMP**

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

R

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

# Unary Relational Operations: SELECT and PROJECT

- ***Sequences of Operations and the RENAME Operation***
- For a SELECT operation with no renaming,
  - the resulting relation names of the attributes are the same as those in the original relation and in the same order.
- For a PROJECT operation with no renaming,
  - the resulting relation has the same attribute names as those in the projection list and in the same order in which they appear in the list.

# Unary Relational Operations: SELECT and PROJECT

- ***Sequences of Operations and the RENAME Operation***
- RENAME operation
  - can rename either the relation name or the attribute names, or both
  - a unary operator.
- symbol ρ (rho) is used to denote the RENAME operator, S is the new relation name, and B1, B2, … , Bn are the new attribute names.
  - renames both the relation and its attributes, $\rho_{S(B1,\, B2,\, \ldots,\, Bn)}(R)$
  - renames the relation only, $\rho_S(R)$
  - renames the attributes only. $\rho_{(B1,\, B2,\, \ldots,\, Bn)}(R)$

# Relational Algebra Operations from Set Theory

- ***The UNION, INTERSECTION, and MINUS Operations***
- Several set theoretic operations are used to merge the elements of two sets in various ways, including UNION, INTERSECTION, and SET DIFFERENCE (also called MINUS or EXCEPT).
- These are **binary operations; that is, each is applied to two sets (of tuples).**
- Two relations R(A1, A2, ... , An) and S(B1, B2, ... , Bn) are said to be **union compatible** (or type compatible) if they have the same degree n and if dom(Ai) = dom(Bi) for 1 ≤ i ≤ n.
- **This means that the two relations have the same number of attributes and each corresponding pair of attributes has the same domain.**

# 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"

# Relational Algebra Operations from Set Theory: SET DIFFERENCE

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

# Relational Algebra Operations from Set Theory

- *The UNION, INTERSECTION, and MINUS Operations*
- ■ UNION: The result of this operation, denoted by 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.
- ■ INTERSECTION: The result of this operation, denoted by R ∩ S, is a relation that includes all tuples that are in both R and S.
- ■ SET DIFFERENCE (or MINUS): The result of this operation, denoted by R – S, is a relation that includes all tuples that are in R but not in S.

# Relational Algebra Operations from Set Theory

- *The UNION, INTERSECTION, and MINUS Operations*
- For example,
  - to retrieve the Social Security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5

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

**RESULT1**

| Ssn |
|---|
| 123456789 |
| 333445555 |
| 666884444 |
| 453453453 |

**RESULT2**

| Ssn |
|---|
| 333445555 |
| 888665555 |

**RESULT**

| Ssn |
|---|
| 123456789 |
| 333445555 |
| 666884444 |
| 453453453 |
| 888665555 |

# Relational Algebra Operations from Set Theory

- ***The UNION, INTERSECTION, and MINUS Operations***
- The relation RESULT1 has the Ssn of all employees who work in department 5, whereas RESULT2 has the Ssn of all employees who directly supervise an employee who works in department 5.
- The UNION operation produces the tuples that are in either RESULT1 or RESULT2 or both **while eliminating any duplicates.**
- Thus, the Ssn value ‘333445555’ appears only once in the result.

# Relational Algebra Operations from Set Theory

- ***The UNION, INTERSECTION, and MINUS Operations***
- **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 because both are **also associative operations**; that is,
- R ∪ (S ∪ T ) = (R ∪ S) ∪ T
- and (R ∩ S) ∩ T = R ∩ (S ∩ T)

**Figure 8.4**
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 |

# Relational Algebra Operations from Set Theory

- ***The UNION, INTERSECTION, and MINUS Operations***
- The MINUS operation is not commutative; that is, in general,
- R − S ≠ S − R
- In SQL, there are three operations—UNION, INTERSECT, and EXCEPT.
- In addition, there are multiset operations (UNION ALL, INTERSECT ALL, and EXCEPT ALL) that do not eliminate duplicates.

**Figure 8.4**
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 |

# Relational Algebra Operations from Set Theory

- ***The CARTESIAN PRODUCT (CROSS PRODUCT) Operation***
- also known as CROSS PRODUCT or CROSS JOIN
- denoted by ×.
- This is a **binary set operation**
- no requirement for relations to be union compatible
- In its binary form, this set operation produces a new element by
  - by combining every member (tuple) from one relation (set) with every member (tuple) from the other relation (set).
- Doing R x S
  - Consider R having *n tuples* and S having *m tuples*
  - And R having *i attributes* & S having *j attributes*
  - The resulting relation will contain n*m tuples & i+j attributes

# Relational Algebra Operations from Set Theory

- ***The CARTESIAN PRODUCT (CROSS PRODUCT) Operation***
- **Query: retrieve a list of names of each female employee's dependents.**

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

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

**FEMALE_EMPS**

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

**EMPNAMES**

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

**EMP_DEPENDENTS**

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

**ACTUAL_DEPENDENTS**

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

**RESULT**

| Fname | Lname | Dependent_name |
|---|---|---|
| Jennifer | Wallace | Abner |

# Binary Relational Operations: JOIN and DIVISION

- ***The JOIN Operation***
- The **JOIN operation**, denoted by: $\bowtie$
- is used to combine related tuples from two relations into single “longer” tuples.
- The general form of a JOIN operation on two relations5 R(A1, A2, ... , An) and S(B1, B2, ... , Bm) is:

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

- **In JOIN, only combinations of tuples satisfying the join condition appear in the result, whereas in the CARTESIAN PRODUCT all combinations of tuples are included in the result.**
- The join condition is specified on attributes from the two relations R and S and is evaluated for each combination of tuples.
- Each tuple combination for which the join condition evaluates to TRUE is included in the resulting relation Q as a single combined tuple
  - **So a join operation is basically the cartesian product of 2 relations followed by a select operation**

# Binary Relational Operations: JOIN and DIVISION

- ***The JOIN Operation***
- **Query: 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 Mgr_ssn value in the department tuple.
- JOIN operation and then project the result over the necessary attributes

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

- Where,
  - Mgr_ssn is a foreign key of the DEPARTMENT relation
  - that references Ssn, the primary key of the EMPLOYEE relation.

# Binary Relational Operations: JOIN and DIVISION

- ***The JOIN Operation : THETA JOIN (Ɵ)***
- **A JOIN operation with any general join condition is called a THETA JOIN.**
- Suppose we have two relation R & S having following attributes.

| R | A1 | A2 |
|---|---|---|
| | 20 | 25 |
| | 80 | 40 |

| S | B1 |
|---|---|
| | 50 |
| | 35 |

- General way to write is:
  - $R \bowtie_{Ai\ \theta\ Bj} S$
- θ (theta) is one of the comparison operators {=, <, ≤, >, ≥, ≠}.
- Consider I want to do $R \bowtie_{A2\ >\ B1} S$
- Tuples whose join attributes are NULL or for which the join condition is FALSE do not appear in the result.

| A1 | A2 | B1 |
|---|---|---|
| 20 | 25 | 50 |
| 20 | 25 | 35 |
| 80 | 40 | 50 |
| 80 | 40 | 35 |

| A1 | A2 | B1 |
|---|---|---|
| 80 | 40 | 35 |

# Binary Relational Operations: JOIN and DIVISION

- *Variations of JOIN: The EQUIJOIN and NATURAL JOIN*
- **EQUI JOIN:**
- Most commonly used join is EQUI join
- Equi join uses ‘=‘ as its comparison operator.
- result of an EQUIJOIN contains one or more pairs of attributes that have identical values in every tuple.
- Here , The values of the attributes Mgr_ssn and Ssn are identical in every tuple of DEPT_MGR (the EQUIJOIN result) because the equality join condition specified on these two attributes requires the values to be identical in every tuple in the result.

$$Dept\_Mgr \leftarrow DEPARTMENT \bowtie_{Mgr\_SSN = SSN} EMPLOYEE$$

# Binary Relational Operations: JOIN and DIVISION

- ***Variations of JOIN: The EQUIJOIN and NATURAL JOIN***
- **NATURAL JOIN:**
- NATURAL JOIN denoted by *
- Condition:
  - the two join attributes (or each pair of join attributes) have the same name in both relations.
  - If this is not the case, a renaming operation is applied first.
- If the attributes on which the natural join is specified already have the same names in both relations, **renaming is unnecessary**.

# Binary Relational Operations: JOIN and DIVISION

- ***Variations of JOIN: The EQUIJOIN and NATURAL JOIN***
- **NATURAL JOIN:**
- The attribute Dnum is called the join attribute for the NATURAL JOIN operation,
- As it can be the only attribute which is repeated in both relations.

PROJECT

| PID | PName | DNum |
|---|---|---|
| 101 | ProjectX | 1 |
| 102 | ProjectY | 2 |
| 103 | ProjectZ | 2 |

DEPARTMENT

| DNo | Mgr_SSN |
|---|---|
| 1 | 553621425 |
| 2 | 996856974 |

$$\text{Proj\_Dept} \leftarrow \text{PROJECT} * \rho_{(\text{DNum, Mgr\_SSN})}(\text{DEPARTMENT})$$

Proj_Dept

| PID | PName | DNum | Mgr_SSN |
|---|---|---|---|
| 101 | ProjectX | 1 | 553621425 |
| 102 | ProjectY | 2 | 996856974 |
| 103 | ProjectZ | 2 | 996856974 |

# Binary Relational Operations: JOIN and DIVISION

- ***Variations of JOIN: The EQUIJOIN and NATURAL JOIN***
- DEPARTMENT :

| Dname | Dnumber | Mgr_ssn | Mgr_start_date |
|---|---|---|---|

- DEPT_LOCATIONS:

| Dnumber | Dlocation |
|---|---|

- to apply a natural join on DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write:

$$\text{DEPT\_LOCS} \leftarrow \text{DEPARTMENT} * \text{DEPT\_LOCATIONS}$$

- The resulting relation combines each department with its locations and has one tuple for each location.

(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 |

# Binary Relational Operations: JOIN and DIVISION

- ***Variations of JOIN: The EQUIJOIN and NATURAL JOIN***
- if no combination of tuples satisfies the join condition,
  - the result of a JOIN is an **empty relation with zero tuples**.
- If there is no join condition,
  - The result is the combinations of tuples of participating relations i.e., **CARTESIAN PRODUCT, also called CROSS PRODUCT or CROSS JOIN**.
- A join operation can be defined as
  - a combination of **CARTESIAN PRODUCT and SELECTION**.

# Binary Relational Operations: JOIN and DIVISION

- ***The JOIN Operation***

- Cartesian product & selection operation :

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

- These two operations can be replaced with a single JOIN operation as follows:

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

# Binary Relational Operations: JOIN and DIVISION

- ***Variations of JOIN: The EQUIJOIN and NATURAL JOIN***
- Multi-way Equi Join can also be performed as follow:

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

- 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's manager.

# Binary Relational Operations: JOIN and DIVISION

- ***The DIVISION Operation***
- The **DIVISION** operation, denoted by ÷, is useful for a special kind of query that sometimes occurs in database applications.
- An example is : **Retrieve the names of employees who work on all the projects that 'John Smith' works on.**
- To express this query using the DIVISION operation, proceed as follows.
- **STEPS:**
- Retrieve the list of project numbers that 'John Smith' works on in the intermediate relation SMITH_PNOS:

**EMPLOYEE**

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno |
|---|---|---|---|---|---|---|---|---|---|

**WORKS_ON**

| Essn | Pno | Hours |
|---|---|---|

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

$$SMITH\_PNOS \leftarrow \pi_{Pno}(WORKS\_ON \bowtie_{Essn=Ssn} SMITH)$$

# Binary Relational Operations: JOIN and DIVISION

- ***The DIVISION Operation***
- Next, create a relation that includes a tuple <Pno, Essn> whenever the employee whose Ssn is Essn works on the project whose number is Pno in the intermediate relation SSN_PNOS:

$$SSN\_PNOS \leftarrow \pi_{Essn,\ Pno}(WORKS\_ON)$$

- Finally, apply the DIVISION operation to the two relations, which gives the desired employees' Social Security numbers:

$$SSNS(Ssn) \leftarrow SSN\_PNOS \div SMITH\_PNOS$$
$$RESULT \leftarrow \pi_{Fname,\ Lname}(SSNS * EMPLOYEE)$$

**Figure 8.8**
The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.

**(a)**

**SSN_PNOS**

| Essn | Pno |
|---|---|
| 123456789 | 1 |
| 123456789 | 2 |
| 666884444 | 3 |
| 453453453 | 1 |
| 453453453 | 2 |
| 333445555 | 2 |
| 333445555 | 3 |
| 333445555 | 10 |
| 333445555 | 20 |
| 999887777 | 30 |
| 999887777 | 10 |
| 987987987 | 10 |
| 987987987 | 30 |
| 987654321 | 30 |
| 987654321 | 20 |
| 888665555 | 20 |

**SMITH_PNOS**

| Pno |
|---|
| 1 |
| 2 |

**SSNS**

| Ssn |
|---|
| 123456789 |
| 453453453 |

**(b)**

**R**

| A | B |
|---|---|
| a1 | b1 |
| a2 | b1 |
| a3 | b1 |
| a4 | b1 |
| a1 | b2 |
| a3 | b2 |
| a2 | b3 |
| a3 | b3 |
| a4 | b3 |
| a1 | b4 |
| a2 | b4 |
| a3 | b4 |

**S**

| A |
|---|
| a1 |
| a2 |
| a3 |

**T**

| B |
|---|
| b1 |
| b4 |

# Binary Relational Operations: JOIN and DIVISION

- ***A Complete Set of Relational Algebra Operations***
- It has been shown that the set of relational algebra operations {σ, π, ∪, ρ, –, ×} is a complete set;
- that is, any of the other original relational algebra operations can be expressed as a sequence of operations from this set.
- For example, the INTERSECTION operation can be expressed by using UNION and MINUS as follows:

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

- JOIN operation can be specified as a CARTESIAN PRODUCT followed by a SELECT operation.

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

# **Table 8.1** Operations of Relational Algebra Summary

**Table 8.1** Operations of Relational Algebra

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

# Table 8.1 Operations of Relational Algebra Summary(continued)

**Table 8.1** Operations of Relational Algebra

| OPERATION | PURPOSE | NOTATION |
|---|---|---|
| UNION | Produces a relation that includes all the tuples in $R_1$ or $R_2$ or both $R_1$ and $R_2$; $R_1$ and $R_2$ must be union compatible. | $R_1 \cup R_2$ |
| INTERSECTION | Produces a relation that includes all the tuples in both $R_1$ and $R_2$; $R_1$ and $R_2$ must be union compatible. | $R_1 \cap R_2$ |
| DIFFERENCE | Produces a relation that includes all the tuples in $R_1$ that are not in $R_2$; $R_1$ and $R_2$ must be union compatible. | $R_1 - R_2$ |
| CARTESIAN PRODUCT | Produces a relation that has the attributes of $R_1$ and $R_2$ and includes as tuples all possible combinations of tuples from $R_1$ and $R_2$. | $R_1 \times R_2$ |
| DIVISION | Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in $\mathrm{R}_1$ in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$. | $R_1(Z) \div R_2(Y)$ |