

Chapter 5: Relational Algebra

cuu duong than cong . com

cuu duong than cong . com

Contents

- 1 Unary Relational Operations
 - 2 Relational Algebra Operations from Set Theory
 - 3 Binary Relational Operations
 - 4 Additional Relational Operations
 - 5 Brief Introduction to Relational Calculus
-

Contents

- 1 Unary Relational Operations**
 - 2 Relational Algebra Operations from Set Theory
 - 3 Binary Relational Operations
 - 4 Additional Relational Operations
 - 5 Brief Introduction to Relational Calculus
-

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).
- A sequence of relational algebra operations forms a **relational algebra expression**.

Relational Algebra Overview

- Unary Relational Operations:
 - SELECT (symbol: σ (sigma))
 - PROJECT (symbol: π (pi))
 - RENAME (symbol: ρ (rho))
- Relational Algebra Operations from Set Theory:
 - UNION (\cup), INTERSECTION (\cap), DIFFERENCE (or MINUS, $-$)
 - CARTESIAN PRODUCT (\times)
- Binary Relational Operations:
 - JOIN (several variations of JOIN exist)
 - DIVISION
- Additional Relational Operations:
 - OUTER JOINS, OUTER UNION
 - AGGREGATE FUNCTIONS (SUM, COUNT, AVG, MIN, MAX)

COMPANY Database Schema

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 |
|-------------|-----------------------|-----|-------|--------------|
|-------------|-----------------------|-----|-------|--------------|

The following query results refer to this database state

One possible database state for the COMPANY relational database schema

EMPLOYEE

| Fname | Minit | Lname | <u>Ssn</u> | 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 |
| Alicia | J | Zelaya | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 |
| 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 |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 25000 | 987654321 | 4 |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 55000 | NULL | 1 |

The following query results refer to this database state

DEPARTMENT

| Dname | <u>Dnumber</u> | Mgr_ssn | Mgr_start_date |
|----------------|----------------|-----------|----------------|
| Research | 5 | 333445555 | 1988-05-22 |
| Administration | 4 | 987654321 | 1995-01-01 |
| Headquarters | 1 | 888665555 | 1981-06-19 |

DEPT_LOCATIONS

| <u>Dnumber</u> | <u>Dlocation</u> |
|----------------|------------------|
| 1 | Houston |
| 4 | Stafford |
| 5 | Bellaire |
| 5 | Sugarland |
| 5 | Houston |

WORKS_ON

| <u>Essn</u> | <u>Pno</u> | Hours |
|-------------|------------|-------|
| 123456789 | 1 | 32.5 |
| 123456789 | 2 | 7.5 |
| 666884444 | 3 | 40.0 |
| 453453453 | 1 | 20.0 |
| 453453453 | 2 | 20.0 |
| 333445555 | 2 | 10.0 |
| 333445555 | 3 | 10.0 |
| 333445555 | 10 | 10.0 |
| 333445555 | 20 | 10.0 |
| 999887777 | 30 | 30.0 |
| 999887777 | 10 | 10.0 |
| 987987987 | 10 | 35.0 |
| 987987987 | 30 | 5.0 |
| 987654321 | 30 | 20.0 |
| 987654321 | 20 | 15.0 |
| 888665555 | 20 | NULL |

PROJECT

| Pname | <u>Pnumber</u> | Plocation | Dnum |
|-----------------|----------------|-----------|------|
| ProductX | 1 | Bellaire | 5 |
| ProductY | 2 | Sugarland | 5 |
| ProductZ | 3 | Houston | 5 |
| Computerization | 10 | Stafford | 4 |
| Reorganization | 20 | Houston | 1 |
| Newbenefits | 30 | Stafford | 4 |

DEPENDENT

| <u>Essn</u> | <u>Dependent_name</u> | Sex | Bdate | Relationship |
|-------------|-----------------------|-----|------------|--------------|
| 333445555 | Alice | F | 1986-04-05 | Daughter |
| 333445555 | Theodore | M | 1983-10-25 | Son |
| 333445555 | Joy | F | 1958-05-03 | Spouse |
| 987654321 | Abner | M | 1942-02-28 | Spouse |
| 123456789 | Michael | M | 1988-01-04 | Son |
| 123456789 | Alice | F | 1988-12-30 | Daughter |
| 123456789 | Elizabeth | F | 1967-05-05 | Spouse |

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**.

- Examples: [cuu duong than cong . com](http://cuuduongthancong.com)

- Select the EMPLOYEE tuples whose department number is 4:

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

- Select the employee tuples whose salary is greater than \$30,000:

$$\sigma_{SALARY > 30,000} (EMPLOYEE)$$

Unary Relational Operations: SELECT

- In general, the *select* operation is denoted by $\sigma_{\langle \text{selection condition} \rangle}(R)$ where
 - σ (sigma) is used to denote the *select* operator.
 - $\langle \text{selection condition} \rangle$ is a Boolean expression specified on the attributes of relation R.
 - Tuples that make the condition **true** appear in the result of the operation, and tuples that make the condition **false** are discarded from the result of the operation.

Unary Relational Operations: SELECT

■ SELECT Operation Properties

- The relation $S = \sigma_{\langle \text{selection condition} \rangle}(R)$ has the same schema (same attributes) as R .
- SELECT σ is **commutative**:

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

- Because of commutativity property, a cascade (sequence) of SELECT operations may be applied in any order:

$$\begin{aligned} \sigma_{\langle \text{cond1} \rangle}(\sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond3} \rangle}(R))) \\ &= \sigma_{\langle \text{cond2} \rangle}(\sigma_{\langle \text{cond3} \rangle}(\sigma_{\langle \text{cond1} \rangle}(R))) \\ &= \sigma_{\langle \text{cond1} \rangle \text{AND} \langle \text{cond2} \rangle \text{AND} \langle \text{cond3} \rangle}(R) \end{aligned}$$

- 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 .

Example of SELECT operation

R

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 1 | 2 | 7 |
| 8 | 4 | 5 |

$\sigma_{A \leq 4} R$

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 1 | 2 | 7 |

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:

$\pi_{\text{LNAME, FNAME, SALARY}}(\text{EMPLOYEE})$

Unary Relational Operations: PROJECT

- The general form of the *project* operation is:

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

- $\langle \text{attribute list} \rangle$ is the desired list of attributes from relation R

cuu duong than cong . com

- The project operation **removes any duplicate tuples** because the result of the *project* operation *do not allow* duplicate elements.

cuu duong than cong . com

Unary Relational Operations: PROJECT

■ PROJECT Operation Properties

- The number of tuples in the result of projection $\pi_{\langle \text{list} \rangle}(R)$ is always ***less than 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
- $\pi_{\langle \text{list1} \rangle}(\pi_{\langle \text{list2} \rangle}(R)) = \pi_{\langle \text{list1} \rangle}(R)$ as long as *$\langle \text{list2} \rangle$ contains the attributes in $\langle \text{list1} \rangle$*

Example of PROJECT operation

R

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 1 | 2 | 7 |
| 8 | 4 | 5 |

π_{AB} R

| A | B |
|---|---|
| 1 | 2 |
| 4 | 5 |
| 8 | 4 |

Examples of applying SELECT and PROJECT operations

Results of SELECT and PROJECT operations. (a) $\sigma_{(Dno=4 \text{ AND } Salary > 25000) \text{ OR } (Dno=5 \text{ AND } Salary > 30000)}(EMPLOYEE)$.
 (b) $\pi_{Lname, Fname, Salary}(EMPLOYEE)$. (c) $\pi_{Sex, Salary}(EMPLOYEE)$.

(a)

| Fname | Minit | Lname | <u>Ssn</u> | 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 |

(b)

| 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 |

(c)

| Sex | Salary |
|-----|--------|
| M | 30000 |
| M | 40000 |
| F | 25000 |
| F | 43000 |
| M | 38000 |
| M | 25000 |
| M | 55000 |

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.

Single expression versus sequence of relational operations

- 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:

$$\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO}=5}(\text{EMPLOYEE}))$$

- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
 - $\text{DEP5_EMPS} \leftarrow \sigma_{\text{DNO}=5}(\text{EMPLOYEE})$
 - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5_EMPS})$

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).

Unary Relational Operations:

RENAME

- The general RENAME operation ρ can be expressed by any of the following forms:
 - $\rho_S(B_1, B_2, \dots, B_n)(R)$ changes both:
 - the relation name to S , *and*
 - the column (attribute) names to B_1, B_2, \dots, B_n
 - $\rho_S(R)$ changes:
 - the *relation name* only to S
 - $\rho_{(B_1, B_2, \dots, B_n)}(R)$ changes:
 - the *column (attribute) names* only to B_1, B_2, \dots, B_n

Contents

-
- 1 Unary Relational Operations
 - 2 Relational Algebra Operations from Set Theory**
 - 3 Binary Relational Operations
 - 4 Additional Relational Operations**
 - 5 Brief Introduction to Relational Calculus
-

Relational Algebra Operations from Set Theory: UNION

- Binary operation, denoted by \cup .
- The result of $R \cup S$, 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).

Example of the result of a UNION operation

RESULT1

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

RESULT2

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

RESULT

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

Figure 6.3

Result of the UNION operation
 $\text{RESULT} \leftarrow \text{RESULT1} \cup \text{RESULT2}.$

Relational Algebra Operations from Set Theory

- Type Compatibility of operands is required for the binary set operation UNION \cup , (also for INTERSECTION \cap , SET DIFFERENCE $-$).

cuu duong than cong . com

- The resulting relation for $R1 \cup R2$ (also for $R1 \cap R2$, or $R1 - R2$) has the *same attribute names* as the first operand relation $R1$ (by convention).

cuu duong than cong . com

Relational Algebra Operations from Set Theory: INTERSECTION

- INTERSECTION is denoted by \cap .
- The result of the operation $R \cap S$, 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”.

Figure 6.4

The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) $\text{STUDENT} \cup \text{INSTRUCTOR}$. (c) $\text{STUDENT} \cap \text{INSTRUCTOR}$. (d) $\text{STUDENT} - \text{INSTRUCTOR}$. (e) $\text{INSTRUCTOR} - \text{STUDENT}$.

Example to
illustrate the
result of
UNION,
INTERSECT,
DIFFERENCE

(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 |

Some properties of UNION, INTERSECT, and DIFFERENCE

- Notice that both union and intersection are **commutative** operations; that is:
 - $R \cup S = S \cup R$, and $R \cap S = S \cap R$
- Both union and intersection can be treated as n-ary operations applicable to any number of relations because both are **associative** operations:
 - $R \cup (S \cup T) = (R \cup S) \cup T$
 - $(R \cap S) \cap T = R \cap (S \cap T)$
- The minus operation is **not commutative**; that is, in general
 - $R - S \neq S - R$

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT

■ CARTESIAN (or CROSS) PRODUCT Operation

- Denoted by $R(A_1, A_2, \dots, A_n) \times S(B_1, B_2, \dots, B_m)$
- Result is a relation with degree $n + m$ attributes:
 - $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$, in that order.
- Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then $R \times S$ will have $n_R * n_S$ tuples.
- The two operands do **NOT** have to be "type compatible".

Example of CARTESIAN PRODUCT operation

R

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 1 | 2 | 7 |
| 8 | 4 | 5 |

T

| A | D |
|---|---|
| 1 | 5 |
| 3 | 7 |

R × T

| R.A | B | C | T.A | D |
|-----|---|---|-----|---|
| 1 | 2 | 3 | 1 | 5 |
| 1 | 2 | 3 | 3 | 7 |
| 4 | 5 | 6 | 1 | 5 |
| 4 | 5 | 6 | 3 | 7 |
| 1 | 2 | 7 | 1 | 5 |
| 1 | 2 | 7 | 3 | 7 |
| 8 | 4 | 5 | 1 | 5 |
| 8 | 4 | 5 | 3 | 7 |

Contents

-
- 1 Unary Relational Operations
 - 2 Relational Algebra Operations from Set Theory
 - 3 Binary Relational Operations**
 - 4 Additional Relational Operations
 - 5 Brief Introduction to Relational Calculus
-

cuu duong than cong . com

Binary Relational Operations: JOIN

- JOIN Operation (denoted by \bowtie)
 - ❑ 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

Binary Relational Operations: JOIN

- JOIN Operation (denoted by \bowtie)
 - The general form of a join operation on two relations $R(A_1, A_2, \dots, A_n)$ and $S(B_1, B_2, \dots, B_m)$ is:

$$R \bowtie_{\langle \text{join condition} \rangle} S$$

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

Binary Relational Operations: JOIN

- 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.

$\text{DEPT_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRSSN}=\text{SSN}} \text{EMPLOYEE}$

- $\text{MGRSSN} = \text{SSN}$ is the join condition
 - Combines each department record with the employee who manages the department.
 - The join condition can also be specified as:
 $\text{DEPARTMENT.MGRSSN} = \text{EMPLOYEE.SSN}$

COMPANY Database Schema

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 |
|-------------|-----------------------|-----|-------|--------------|
|-------------|-----------------------|-----|-------|--------------|

The following query results refer to this database state

One possible database state for the COMPANY relational database schema

EMPLOYEE

| Fname | Minit | Lname | <u>Ssn</u> | 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 |
| Alicia | J | Zelaya | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 |
| 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 |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 25000 | 987654321 | 4 |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 55000 | NULL | 1 |

DEPARTMENT

| Dname | <u>Dnumber</u> | Mgr_ssn | Mgr_start_date |
|----------------|----------------|-----------|----------------|
| Research | 5 | 333445555 | 1988-05-22 |
| Administration | 4 | 987654321 | 1995-01-01 |
| Headquarters | 1 | 888665555 | 1981-06-19 |

The following query results refer to this database state

DEPARTMENT

| Dname | <u>Dnumber</u> | Mgr_ssn | Mgr_start_date |
|----------------|----------------|-----------|----------------|
| Research | 5 | 333445555 | 1988-05-22 |
| Administration | 4 | 987654321 | 1995-01-01 |
| Headquarters | 1 | 888665555 | 1981-06-19 |

DEPT_LOCATIONS

| <u>Dnumber</u> | <u>Dlocation</u> |
|----------------|------------------|
| 1 | Houston |
| 4 | Stafford |
| 5 | Bellaire |
| 5 | Sugarland |
| 5 | Houston |

WORKS_ON

| <u>Essn</u> | <u>Pno</u> | Hours |
|-------------|------------|-------|
| 123456789 | 1 | 32.5 |
| 123456789 | 2 | 7.5 |
| 666884444 | 3 | 40.0 |
| 453453453 | 1 | 20.0 |
| 453453453 | 2 | 20.0 |
| 333445555 | 2 | 10.0 |
| 333445555 | 3 | 10.0 |
| 333445555 | 10 | 10.0 |
| 333445555 | 20 | 10.0 |
| 999887777 | 30 | 30.0 |
| 999887777 | 10 | 10.0 |
| 987987987 | 10 | 35.0 |
| 987987987 | 30 | 5.0 |
| 987654321 | 30 | 20.0 |
| 987654321 | 20 | 15.0 |
| 888665555 | 20 | NULL |

PROJECT

| Pname | <u>Pnumber</u> | Plocation | Dnum |
|-----------------|----------------|-----------|------|
| ProductX | 1 | Bellaire | 5 |
| ProductY | 2 | Sugarland | 5 |
| ProductZ | 3 | Houston | 5 |
| Computerization | 10 | Stafford | 4 |
| Reorganization | 20 | Houston | 1 |
| Newbenefits | 30 | Stafford | 4 |

DEPENDENT

| <u>Essn</u> | <u>Dependent_name</u> | Sex | Bdate | Relationship |
|-------------|-----------------------|-----|------------|--------------|
| 333445555 | Alice | F | 1986-04-05 | Daughter |
| 333445555 | Theodore | M | 1983-10-25 | Son |
| 333445555 | Joy | F | 1958-05-03 | Spouse |
| 987654321 | Abner | M | 1942-02-28 | Spouse |
| 123456789 | Michael | M | 1988-01-04 | Son |
| 123456789 | Alice | F | 1988-12-30 | Daughter |
| 123456789 | Elizabeth | F | 1967-05-05 | Spouse |

Example of applying the JOIN operation

DEPT_MGR

| Dname | Dnumber | Mgr_ssn | ... | Fname | Minit | Lname | Ssn | ... |
|----------------|---------|-----------|-----|----------|-------|---------|-----------|-----|
| Research | 5 | 333445555 | ... | Franklin | T | Wong | 333445555 | ... |
| Administration | 4 | 987654321 | ... | Jennifer | S | Wallace | 987654321 | ... |
| Headquarters | 1 | 888665555 | ... | James | E | Borg | 888665555 | ... |

Result of the JOIN operation

DEPT_MGR ← DEPARTMENT  EMPLOYEE
MGRSSN=SSN

Example of JOIN operation

R

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 1 | 2 | 7 |
| 8 | 4 | 5 |

R $\bowtie_{R.A > T.A}$ **T**

| R.A | B | C | T.A | D |
|-----|---|---|-----|---|
| 4 | 5 | 6 | 1 | 5 |
| 4 | 5 | 6 | 3 | 7 |
| 8 | 4 | 5 | 1 | 5 |
| 8 | 4 | 5 | 3 | 7 |

T

| A | D |
|---|---|
| 1 | 5 |
| 3 | 7 |

Some properties of JOIN

■ Consider the following JOIN operation:

$$\square R(A_1, A_2, \dots, A_n) \bowtie S(B_1, B_2, \dots, B_m)$$

$R.A_i = S.B_j$

- Result is a relation Q with degree **n + m** attributes:
 - $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$, 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* $r[A_i] = s[B_j]$.
- Hence, if R has n_R tuples, and S has n_S tuples, then the join result will generally have *less than* $n_R * n_S$ tuples.
- Only related tuples (based on the join condition) will appear in the result.

Some properties of JOIN

- The general case of JOIN operation is called a Theta-join: $R \bowtie_{\langle \theta \rangle} S$
- The join condition is called *theta*.
- *Theta* can be any general boolean expression on the attributes of R and S; for example:
 - $R.A_i < S.B_j \text{ AND } (R.A_k = S.B_l \text{ OR } R.A_p < S.B_q)$

Binary Relational Operations:

EQUIJOIN

- 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.

cuu duong than cong . com

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.
- 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

■ Example: $Q \leftarrow R(A,B,C,D) * S(C,D,E)$

- 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:
 - $Q(A,B,C,D,E)$

Example of NATURAL JOIN operation

R

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 1 | 2 | 7 |
| 8 | 4 | 5 |

R * T

| A | B | C | D |
|---|---|---|---|
| 1 | 2 | 3 | 5 |
| 1 | 2 | 7 | 5 |

T

| A | D |
|---|---|
| 1 | 5 |
| 3 | 7 |

Example of NATURAL JOIN operation

(a)

PROJ_DEPT

| Pname | <u>Pnumber</u> | Plocation | Dnum | Dname | Mgr_ssn | Mgr_start_date |
|-----------------|----------------|-----------|------|----------------|-----------|----------------|
| ProductX | 1 | Bellaire | 5 | Research | 333445555 | 1988-05-22 |
| ProductY | 2 | Sugarland | 5 | Research | 333445555 | 1988-05-22 |
| ProductZ | 3 | Houston | 5 | Research | 333445555 | 1988-05-22 |
| Computerization | 10 | Stafford | 4 | Administration | 987654321 | 1995-01-01 |
| Reorganization | 20 | Houston | 1 | Headquarters | 888665555 | 1981-06-19 |
| Newbenefits | 30 | Stafford | 4 | Administration | 987654321 | 1995-01-01 |

(b)

DEPT_LOCS

| Dname | Dnumber | Mgr_ssn | Mgr_start_date | Location |
|----------------|---------|-----------|----------------|-----------|
| Headquarters | 1 | 888665555 | 1981-06-19 | Houston |
| Administration | 4 | 987654321 | 1995-01-01 | Stafford |
| Research | 5 | 333445555 | 1988-05-22 | Bellaire |
| Research | 5 | 333445555 | 1988-05-22 | Sugarland |
| Research | 5 | 333445555 | 1988-05-22 | Houston |

Figure 6.7

Results of two NATURAL JOIN operations.

(a) PROJ_DEPT \leftarrow PROJECT * DEPT.

(b) DEPT_LOCS \leftarrow DEPARTMENT * DEPT_LOCATIONS.

Complete Set of Relational Operations

- The set of operations $\{\sigma, \pi, \cup, -, \bowtie\}$ is called a complete set because any other relational algebra expressions can be expressed by a combination of these five operations.

cuu duong than cong . com

- For example:

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

- $R \bowtie_{\langle \text{join condition} \rangle} S = \sigma_{\langle \text{join condition} \rangle} (R \times S)$

cuu duong than cong . com

Binary Relational Operations:

DIVISION

■ DIVISION Operation

- The division operation is applied to two relations $R(Z) \div S(X)$, where $Z = X \cup Y$ (Y is 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 t_R appear in R with $t_R[Y] = t$, and with $t_R[X] = t_s$ for every tuple t_s in S , i.e., 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 .

Figure 6.8

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 |

Example of the
DIVISION
operation

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

Table 6.1 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 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.

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

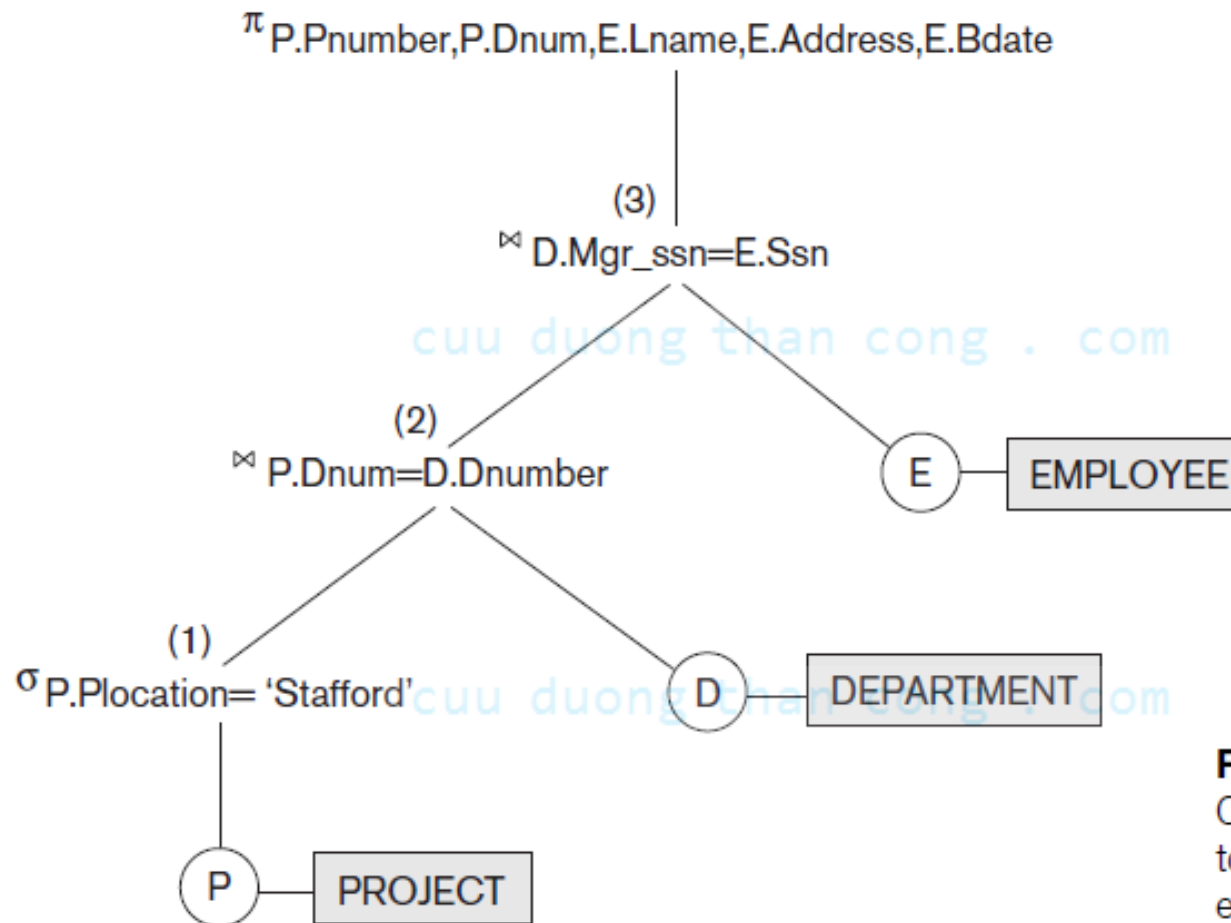


Figure 6.9
 Query tree corresponding
 to the relational algebra
 expression for Q2.

Contents

-
- 1 Unary Relational Operations
 - 2 Relational Algebra Operations from Set Theory
 - 3 Binary Relational Operations
 - 4 Additional Relational Operations**
 - 5 Brief Introduction to Relational Calculus
-

cuu duong than cong . com

Additional Relational Operations

■ Aggregate Functions and Grouping

- ❑ A type of request that cannot be expressed in the basic relational algebra is to specify mathematical **aggregate functions** on collections of values from the database.
- ❑ Examples of such functions include retrieving the average or total salary of all employees or the total number of employee tuples.
- ❑ Common functions applied to collections of numeric values include **SUM, AVERAGE, MAXIMUM, and MINIMUM**. The **COUNT** function is used for counting tuples or values.

Examples of applying aggregate functions and grouping

Figure 6.10

The aggregate function operation.

- a. $\rho_{R(Dno, No_of_employees, Average_sal)}(Dno \bowtie \text{COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE}))$.
- b. $Dno \bowtie \text{COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$.
- c. $\bowtie \text{COUNT Ssn, AVERAGE Salary}(\text{EMPLOYEE})$.

R

(a)

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

(c)

| Count_ssn | Average_salary |
|-----------|----------------|
| 8 | 35125 |

(b)

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

Additional Relational Operations

■ Use of the Functional operator \mathcal{F}

- ❑ $\mathcal{F}_{\text{MAX Salary}}(\text{Employee})$ retrieves the maximum salary value from the Employee relation
- ❑ $\mathcal{F}_{\text{MIN Salary}}(\text{Employee})$ retrieves the minimum Salary value from the Employee relation
- ❑ $\mathcal{F}_{\text{SUM Salary}}(\text{Employee})$ retrieves the sum of the Salary from the Employee relation
- ❑ $\text{DNO } \mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}(\text{Employee})$ groups employees by DNO (department number) and computes the count of employees and average salary per department
- ❑ **Note: count just counts the number of rows, without removing duplicates.**

Additional Relational Operations

■ Recursive Closure Operations

- ❑ Another type of operation that, in general, cannot be specified in the basic original relational algebra is **recursive closure**. This operation is applied to a **recursive relationship**.
- ❑ An example of a recursive operation is to retrieve all SUPERVISEES of an EMPLOYEE *e* at all levels.
- ❑ Although it is possible to retrieve employees at each level and then take their union, we **cannot**, in general, specify a query such as “retrieve the supervisees of ‘James Borg’ at all levels” without utilizing a looping mechanism.
- ❑ The SQL3 standard includes syntax for recursive closure.

Additional Relational Operations

- The OUTER JOIN Operation
 - In NATURAL JOIN and EQUIJOIN, tuples without a *matching* (or *related*) tuple are eliminated from the join result.
 - Tuples with *null* in the join attributes are also eliminated.
 - This amounts to loss of information.
 - A set of operations, called OUTER joins, can be used when we want to keep all the tuples in R, or all those in S, or all those in both relations in the result of the join, regardless of whether or not they have matching tuples in the other relation.
 - Outer Union operations: homework !!

Additional Relational Operations

- The **left outer join** operation keeps every tuple in the **first or left** relation R in $R \bowtie S$; if no matching tuple is found in S, then the attributes of S in the join result are filled or “padded” with **null values**.
- A similar operation, **right outer join**, keeps every tuple in the second or right relation S in the result of $R \bowtie S$.
- A third operation, **full outer join**, denoted by \bowtie , keeps all tuples in both the left and the right relations when no matching tuples are found, padding them with null values as needed.

Additional Relational Operations

- Example: List all *employee names* and also the *name of the departments they manage* if they happen to manage a department (if they do not manage one, we can indicate it with a NULL value)

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

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

The following query results refer to this database state

One possible database state for the COMPANY relational database schema

EMPLOYEE

| Fname | Minit | Lname | <u>Ssn</u> | 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 |
| Alicia | J | Zelaya | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX | F | 25000 | 987654321 | 4 |
| 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 |
| Joyce | A | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX | F | 25000 | 333445555 | 5 |
| Ahmad | V | Jabbar | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX | M | 25000 | 987654321 | 4 |
| James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 55000 | NULL | 1 |

DEPARTMENT

| Dname | <u>Dnumber</u> | Mgr_ssn | Mgr_start_date |
|----------------|----------------|-----------|----------------|
| Research | 5 | 333445555 | 1988-05-22 |
| Administration | 4 | 987654321 | 1995-01-01 |
| Headquarters | 1 | 888665555 | 1981-06-19 |

Additional Relational Operations

RESULT

| Fname | Minit | Lname | Dname |
|----------|-------|---------|----------------|
| John | B | Smith | NULL |
| Franklin | T | Wong | Research |
| Alicia | J | Zelaya | NULL |
| Jennifer | S | Wallace | Administration |
| Ramesh | K | Narayan | NULL |
| Joyce | A | English | NULL |
| Ahmad | V | Jabbar | NULL |
| James | E | Borg | Headquarters |

Figure 6.12

The result of a
LEFT OUTER JOIN
operation.

Example of LEFT OUTER JOIN

R

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 1 | 2 | 7 |
| 8 | 4 | 5 |

R ~~**⋈**~~ **T**

| A | B | C | D |
|---|---|---|------|
| 1 | 2 | 3 | 5 |
| 1 | 2 | 7 | 5 |
| 4 | 5 | 6 | null |
| 8 | 4 | 5 | null |

T

| A | D |
|---|---|
| 1 | 5 |
| 3 | 7 |

Example of RIGHT OUTER JOIN

R

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 1 | 2 | 7 |
| 8 | 4 | 5 |

R \bowtie T

| A | B | C | D |
|---|------|------|---|
| 1 | 2 | 3 | 5 |
| 1 | 2 | 7 | 5 |
| 3 | null | null | 7 |

T

| A | D |
|---|---|
| 1 | 5 |
| 3 | 7 |

Example of FULL OUTER JOIN

R

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 1 | 2 | 7 |
| 8 | 4 | 5 |

~~R~~ ~~∩~~ ~~T~~

| A | B | C | D |
|---|------|------|------|
| 1 | 2 | 3 | 5 |
| 1 | 2 | 7 | 5 |
| 4 | 5 | 6 | null |
| 8 | 4 | 5 | null |
| 3 | null | null | 7 |

T

| A | D |
|---|---|
| 1 | 5 |
| 3 | 7 |

Contents

- 1 Unary Relational Operations
 - 2 Relational Algebra Operations from Set Theory
 - 3 Binary Relational Operations
 - 4 Additional Relational Operations
 - 5 Brief Introduction to Relational Calculus**
-

cuu duong than cong . com

Brief Introduction to Relational Calculus

- A **relational calculus** expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in **tuple calculus**) or over columns of the stored relations (in **domain calculus**).
- In a calculus expression, there is *no order of operations* to specify how to retrieve the query result—a calculus expression specifies only what information the result should contain. This is the main distinguishing feature between relational algebra and relational calculus.
- Relational calculus is considered to be a **nonprocedural** language. This differs from relational algebra, where we must write a *sequence of operations* to specify a retrieval request; hence relational algebra can be considered as a **procedural** way of stating a query.

Brief Introduction to Relational Calculus

- The tuple relational calculus is based on specifying a number of **tuple variables**. Each tuple variable usually *ranges over* a particular database relation, meaning that the variable may take as its value any individual tuple from that relation.
- A simple tuple relational calculus query is of the form **$\{t \mid \text{COND}(t)\}$** where t is a tuple variable and $\text{COND}(t)$ is a conditional expression involving t .
 - **Example:** To find the first and last names of all employees whose salary is above \$50,000, we can write the following tuple calculus expression:
 $\{t.\text{FNAME}, t.\text{LNAME} \mid \text{EMPLOYEE}(t) \text{ AND } t.\text{SALARY} > 50000\}$
 - The condition $\text{EMPLOYEE}(t)$ specifies that the **range relation** of tuple variable t is EMPLOYEE . The first and last name ($\pi_{\text{FNAME}, \text{LNAME}}$) of each EMPLOYEE tuple t that satisfies the condition $t.\text{SALARY} > 50000$ ($\sigma_{\text{SALARY} > 50000}$) will be retrieved.

Brief Introduction to Relational Calculus

- Two special symbols called **quantifiers** can appear in formulas; these are the **universal quantifier** (\forall) and the **existential quantifier** (\exists).
cuu duong than cong . com
- Informally, a tuple variable t is bound if it is quantified, meaning that it appears in an ($\forall t$) or ($\exists t$) clause; otherwise, it is **free**.
cuu duong than cong . com

Brief Introduction to Relational Calculus

- **Example 1:** retrieve the name and address of all employees who work for the 'Research' department.

$\{t.FNAME, t.LNAME, t.ADDRESS \mid EMPLOYEE(t) \text{ and } (\exists d) (DEPARTMENT(d) \text{ and } d.DNAME='Research' \text{ and } d.DNUMBER=t.DNO) \}$

cuu duong than cong . com

Brief Introduction to Relational Calculus

- **Example 2:** find the names of employees who work on *all* the projects controlled by department number 5.

$\{e.LNAME, e.FNAME \mid EMPLOYEE(e) \text{ and } ((\forall x) (\text{not}(\text{PROJECT}(x)) \text{ or } \text{not}(x.DNUM=5))$

$\text{OR } ((\exists w)(\text{WORKS_ON}(w) \text{ and } w.ESSN=e.SSN \text{ and } x.PNUMBER=w.PNO))))\}$

- Details: [1] Chapter 6

Brief Introduction to Relational Calculus

- Another variation of relational calculus called the domain relational calculus, or simply, **domain calculus** is equivalent to tuple calculus and to relational algebra.
- QBE (Query-By-Example): see Appendix D
- Domain calculus differs from tuple calculus in the *type of variables* used in formulas: rather than having variables range over tuples, the variables range over single values from domains of attributes. To form a relation of degree n for a query result, we must have n of these **domain variables** - one for each attribute.
- An expression of the domain calculus is of the form $\{x_1, x_2, \dots, x_n \mid \text{COND}(x_1, x_2, \dots, x_n, x_{n+1}, x_{n+2}, \dots, x_{n+m})\}$, where:
 - $x_1, x_2, \dots, x_n, x_{n+1}, x_{n+2}, \dots, x_{n+m}$ are domain variables that range over domains (of attributes)
 - COND is a **condition** or **formula** of the domain relational calculus.

Brief Introduction to Relational Calculus

- Example: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

$\{uv \mid (\exists q) (\exists r) (\exists s) (\exists t) (\exists w) (\exists x) (\exists y) (\exists z)$
 $(\text{EMPLOYEE}(qrstuvwxyz) \text{ and } q=\text{'John'} \text{ and } r=\text{'B'} \text{ and } s=\text{'Smith'})\}$

Summary

- 1 Unary Relational Operations
 - 2 Relational Algebra Operations from Set Theory
 - 3 Binary Relational Operations
 - 4 Additional Relational Operations
 - 5 Brief Introduction to Relational Calculus
-

cuu duong than cong . com

Q & A

cuu duong than cong . com

Exercise 1

- **Using relational algebra:** retrieve the *name and address of all employees who work for the 'Research' department*

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

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

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

Exercise 2

- **Using relational algebra:** For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

```
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 ←  $\pi_{Pnumber, Dnum, Lname, Address, Bdate}(PROJ\_DEPT\_MGRS)$ 
```

Exercise 3

- **Using relational algebra:** Find the names of employees who work on *all* the projects controlled by department number 5.

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

cuu duong than cong . com

Exercise 4

- **Using relational algebra:** List the names of all employees with two or more dependents.

```
T1(Ssn, No_of_dependents) ←  $\text{Essn} \int \text{COUNT Dependent\_name}(\text{DEPENDENT})$   
T2 ←  $\sigma_{\text{No\_of\_dependents} > 2}(T1)$   
RESULT ←  $\pi_{\text{Lname, Fname}}(T2 \times \text{EMPLOYEE})$ 
```

cuu duong than cong . com

Exercise 5

- **Using relational algebra:** Retrieve the names of employees who have no dependents.

```
ALL_EMPS  $\leftarrow \pi_{Ssn}(EMPLOYEE)$   
EMPS_WITH_DEPS(Ssn)  $\leftarrow \pi_{E, Ssn}(DEPENDENT)$   
EMPS_WITHOUT_DEPS  $\leftarrow (ALL\_EMPS - EMPS\_WITH\_DEPS)$   
RESULT  $\leftarrow \pi_{Lname, Fname}(EMPS\_WITHOUT\_DEPS \times EMPLOYEE)$ 
```

cuu duong than cong . com

Review questions

- 1) List the operations of relational algebra and the purpose of each.
- 2) What is union compatibility? Why do the UNION, INTERSECTION, and DIFFERENCE operations require that the relations on which they are applied be union compatible?
- 3) How are the OUTER JOIN operations different from the INNER JOIN operations? How is the OUTER UNION operation different from UNION?
- 4) In what sense does relational calculus differ from relational algebra, and in what sense are they similar?