Database Management Systems

ICT1212

The Relational Algebra

Department of ICT
Faculty of Technology
University of Ruhuna

Lecture 7

Example : COMPANY

Schema diagram for the COMPANY relational database schema; the primary keys are underlined.

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Example : COMPANY

EMPLOYEE

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
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	Dnumber	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

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	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	Pnumber	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

Essn	Dependent_name	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

Relational Algebra

- The basic **set of operations** for the relational model is known as the relational algebra. These operations enable a user to specify basic retrieval requests as relational algebra operations

- The result of a retrieval is a new relation, which may have been formed from one or more relations. The algebra operations thus produce new relations, which can be further manipulated using operations of the same algebra

- A sequence of relational algebra operations forms a relational algebra expression, whose result will also be a relation that represents the result of a database query (or retrieval request)

Relational Algebra

The relational algebra is very important for several reasons

- It provides a formal foundation for relational model operations
- It is used as a basis for implementing and optimizing queries in the query processing and optimization modules that are integral parts of relational database management systems (RDBMSs)
- Its concepts are incorporated into the SQL standard query language for RDBMSs, the core operations and functions in the internal modules of most relational systems are based on relational algebra operations

Relational Algebra

The relational operations can be divided into two groups

- Set operations
  - UNION, INTERSECTION, SET DIFFERENCE, and CARTESIAN PRODUCT (also known as CROSS PRODUCT)
- Operations developed specifically for relational databases
  - SELECT, PROJECT, and JOIN

Relational Algebra

- Unary Operations
  - Operate on single relations
- Binary Operations
  - Operate on two tables by combining related tuples (records) based on join conditions
- Aggregate Functions
  - Operations that can summarize data from the tables, as well as additional types of JOIN and UNION operations

Unary Relational Operations

- ## SELECT Operation

SELECT operation is used to select a *subset* of the tuples from a relation that satisfy a **selection condition**. It is a filter that keeps only those tuples that satisfy a qualifying condition – those satisfying the condition are selected while others are discarded.

Example:

To select the EMPLOYEE tuples whose department number is 04 or those whose salary is greater than \$30,000 the following notation is used:

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

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

In general, 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 specified on the attributes of relation R

Unary Relational Operations

SELECT Operation Properties

- The SELECT operation
  $\sigma_{\text{<selection condition>}}(R)$
  produces a relation S that has the same schema as R

- The **degree** of the relation resulting from a SELECT operation—its number of attributes—is the same as the degree of R.

- The SELECT operation is commutative
  $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(R)) = \sigma_{\text{<condition2>}}(\sigma_{\text{< condition1>}}(R))$

- A cascaded SELECT operation may be applied in any order
  $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(\sigma_{\text{<condition3>}}(R))$
  $= \sigma_{\text{<condition2>}}(\sigma_{\text{< condition3>}}(\sigma_{\text{< condition1>}}(R)))$

- A cascaded SELECT operation may be replaced by a single selection with a conjunction of all the conditions
  $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(\sigma_{\text{<condition3>}}(R))$
  $= \sigma_{\text{<condition1> AND < condition2> AND < condition3>}}(R)))$

Unary Relational Operations

SELECT Operation

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

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

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

- **PROJECT Operation**

This operation selects certain columns from the table and discards the other columns. The PROJECT creates a vertical partitioning – one with the needed columns (attributes) containing results of the operation and other containing the discarded Columns.

Example:

To list each employee's first and last name and salary, the following is used:

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

The general form of the project operation is

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

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.

The project operation removes any duplicate tuples, so the result of the project operation is a set of distinct tuples(duplicate elimination) and hence a valid relation.

Unary Relational Operations

PROJECT Operation Properties

- The result of the PROJECT operation has only the attributes specified in <attribute list> in the same order as they appear in the list. Hence, its **degree** is equal to the number of attributes in <attribute list>

- The number of tuples in the result of projection

$$\pi_{\textbf{<list>}}(R)$$

  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 is equal to the number of tuples in R.

- $\pi_{\textbf{<list1>}}(\pi_{\textbf{<list2>}}(R)) = \pi_{\textbf{<list1>}}(R)$

  as long as <list2> contains the attributes in <list1>

- It is also note that commutativity does not hold on PROJECT

Unary Relational Operations

PROJECT Operation

Example:

- To select the sex and salary for all employees, we can specify the following PROJECT operation

$$\pi_{Sex,\ Salary}(EMPLOYEE)$$

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

Unary Relational Operations

Rename Operation

- 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(**in-line expression**), 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.

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:

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

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 Operation

The rename operator is ρ

The general Rename operation can be expressed by any of the following forms:

- $\rho_{S\,(B_1, B_2, \ldots, B_n)}(R)$ is a renamed relation S based on R with column names $B_1, B_1, \ldots.. B_n$.
- $\rho_S(R)$ is a renamed relation S based on R (which does not specify column names).
- $\rho_{(B_1, B_2, \ldots, B_n)}(R)$ is a renamed relation with column names B1, B1, …..Bn which does not specify a new relation name.

Unary Relational Operations

Rename Operation

- **Example:**

1. **SELECT** E.Fname, E.Lname, E.Salary
   FROM EMPLOYEE **AS** E
   WHERE E.Dno=5

2. **SELECT** E.Fname **AS** First_name**,** E.Lname **AS** Last_name**,** E.Salary **AS** Salary
   FROM EMPLOYEE **AS** E
   WHERE E.Dno=5

Unary Relational Operations

Rename Operation

Example:

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

Fname	Lname	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

Example:

$\text{TEMP} \leftarrow \sigma_{\text{Dno}=5}(\text{EMPLOYEE})$

$R(\text{First\_name, Last\_name, Salary}) \leftarrow \pi_{\text{Fname, Lname, Salary}}(\text{TEMP})$

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

Relational Algebra Operations from Set Theory

- **UNION Operation**

The result of this operation, denoted by $R \cup S$, 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.

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, we can use the union operation as follows

$\mathbf{DEP5\_EMPS} \leftarrow \sigma_{\mathbf{DNO=5}}\ \mathbf{(EMPLOYEE)}$

$\mathbf{RESULT1} \leftarrow \pi_{\mathbf{SSN}}\mathbf{(DEP5\_EMPS)}$

$\mathbf{RESULT2(SSN)} \leftarrow \pi_{\mathbf{SUPERSSN}}\mathbf{(DEP5\_EMPS)}$

$\mathbf{RESULT} \leftarrow \mathbf{RESULT1} \cup \mathbf{RESULT2}$

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

The two operands must be “type compatible”.

Relational Algebra Operations from Set Theory

- **UNION Example**

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Relational Algebra Operations from Set Theory

- **Union Compatibility/Type Compatibility**
  - Two relations $R(A_1, A_2, ..., A_n)$ and $S(B_1, B_2, ..., B_n)$ are said to be **union compatible** (or **type compatible**) if they have the same degree n and if $dom(A_i) = dom(B_i)$ 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.

  - The resulting relation for $R_1 \cup R_2$ has the same attribute names as the *first* operand relation R_1 (by convention).

Data Set for UNION,INTERSECTION and MINUS

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

Relational Algebra Operations from Set Theory

- **INTERSECTION Operation**

The result of this operation, denoted by $R \cap S$, is a relation that includes all tuples that are in both R and S. The two operands must be "type compatible"

Example:

The result of the intersection operation STUDENT ∩ INSTRUCTOR (figure below) includes only those who are both students and instructors.

Relational Algebra Operations from Set Theory

- **INTERSECTION Example**

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

Fn	Ln
Susan	Yao
Ramesh	Shah

Relational Algebra Operations from Set Theory

- **Set Difference (or MINUS) Operation**

The result of this operation, denoted by R - S, is a relation that includes all tuples that are in R but not in S. The two operands must be "type compatible".

Example:

Below figures shows the names of students who are not instructors(STUDENT − INSTRUCTOR), and the names of instructors who are not students(INSTRUCTOR − STUDENT).

Relational Algebra Operations from Set Theory

- **Set Difference (or MINUS) Operation**

STUDENT − INSTRUCTOR

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR − STUDENT

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

Relational Algebra Operations from Set Theory

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

  $$\mathbf{R \cup S = S \cup R, \text{ and } R \cap S = S \cap R}$$

- Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative operations;* that is

  $$\mathbf{R \cup (S \cup T) = (R \cup S) \cup T, \text{ and } (R \cap S) \cap T = R \cap (S \cap T)}$$

- The minus operation is *not commutative*; that is, in general

  $$\mathbf{R - S \neq S - R}$$

Relational Algebra Operations from Set Theory

- **CARTESIAN (or CROSS PRODUCT) Operation**
  - This operation is used to combine tuples from two relations in a combinatorial fashion. In general, the result of

    $R(A_1, A_2, \ldots, A_n) \times S(B_1, B_2, \ldots, B_m)$

    is a relation Q with degree n + m attributes

    $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ in that order.
  - The resulting relation Q has one tuple for each combination of tuples → one from R and one from S.
  - Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then $| R \times S |$ will have $\mathbf{n_R * n_S}$ tuples.
  - The two operands do NOT have to be "type compatible"

Relational Algebra Operations from Set Theory

- **CARTESIAN (or CROSS PRODUCT) Operation**

Example:

suppose that we want to retrieve a list of names of each female employee's dependents

$$\mathbf{FEMALE\_EMPS} \leftarrow \sigma_{Sex='F'}(\mathbf{EMPLOYEE})$$

$$\mathbf{EMPNAMES} \leftarrow \pi_{Fname,\ Lname,\ Ssn}(\mathbf{FEMALE\_EMPS})$$

$$\mathbf{EMP\_DEPENDENTS} \leftarrow \mathbf{EMPNAMES} \times \mathbf{DEPENDENT}$$

$$\mathbf{ACTUAL\_DEPENDENTS} \leftarrow \sigma_{Ssn=Essn}(\mathbf{EMP\_DEPENDENTS})$$

$$\mathbf{RESULT} \leftarrow \pi_{Fname,\ Lname,\ Dependent\_name}(\mathbf{ACTUAL\_DEPENDENTS})$$

Relational Algebra Operations from Set Theory

- **CARTESIAN (or 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 Operation**
  - The sequence of CARTESIAN product followed by SELECT is used quite commonly to identify and select related tuples from two relations, a special operation, called JOIN. It is denoted by a $\bowtie$
  - This operation is very important for any relational database with more than a single relation, because it allows us to process relationships among relations.
  - The general form of a join operation on two relations R(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is:

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

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

Binary Relational Operations

- **JOIN Operation**

Example:

suppose that we want to retrieve the name of the manager of each department

$$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_ssn=Ssn}} \text{EMPLOYEE}$$

$$\text{RESULT} \leftarrow \pi_{\text{Dname, Lname, Fname}}(\text{DEPT\_MGR})$$

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	...

Binary Relational Operations

- **JOIN Operation**
  - The result of the JOIN is a relation *Q* with $n + m$ attributes $Q(A1, A2, \ldots, An, B1, B2, \ldots, Bm)$ in that order;
  - *Q* has one tuple for each combination of tuples—one from *R* and one from *S*—*whenever the combination satisfies the join condition*. This is the main difference between CARTESIAN PRODUCT and JOIN.
  - 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.
  - A general join condition is of the form

    <condition> **AND** <condition> **AND**...**AND** <condition>

    where each <condition> is of the form

    $$Ai\ \theta\ Bj,$$

    Ai is an attribute of *R*, *Bj* is an attribute of *S*, *Ai* and *Bj* have the same domain, and θ (theta) is one of the comparison operators

    $\{=, <, \leq, >, \geq, \neq\}$.
  - A JOIN operation with such a general join condition is called a **THETA JOIN**.
  - Tuples whose join attributes are NULL or for which the join condition is FALSE *do not* appear in the result.

Binary Relational Operations

- **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 EQUIJOIN.

- **NATURAL JOIN Operation**

  Because one of each pair of attributes with identical values is superfluous, a new operation 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.

Binary Relational Operations

- **NATURAL JOIN Operation**

1. PROJ_DEPT ← PROJECT * ρ(Dname, Dnum, Mgr_ssn, Mgr_start_date)(DEPARTMENT)

OR

DEPT ← ρ(Dname, Dnum, Mgr_ssn, Mgr_start_date)(DEPARTMENT)

PROJ_DEPT ← PROJECT * DEPT

2.DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS

PROJ_DEPT

Pname	Pnumber	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

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

Complete Set of Relational Operations

- The set of operations including **select σ, project π , union $\cup$, set difference - , 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.

- **Example**:

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

2. $R \bowtie_{<join\ condition>} S = \sigma_{<join\ condition>} (R \times S)$

Binary Relational Operations

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

Binary Relational Operations

- **DIVISION Operation**

Example:

- *Retrieve the names of employees who work on* ***all*** *the projects that 'John Smith' works on*

SMITH ← σFname='John' AND Lname='Smith'(EMPLOYEE)

SMITH_PNOS ← πPno(WORKS_ON ⋈ Essn=SsnSMITH)

SSN_PNOS ← πEssn, Pno(WORKS_ON)

SSNS(Ssn) ← SSN_PNOS ÷ SMITH_PNOS

RESULT ← πFname, Lname(SSNS * EMPLOYEE)

Binary Relational Operations

DIVISION Operation

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

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

Recap of Relational Algebra Operations

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

Additional Relational Operations

- **Generalized Projection**
  - The generalized projection operation extends the projection operation by allowing functions of attributes to be included in the projection list. The generalized form can be expressed as:

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

  - where $F1$, $F2$, ..., Fn are functions over the attributes in relation R and may involve arithmetic operations and constant values.
  - This operation is helpful when developing reports where computed values have to be produced in the columns of a query result.

Additional Relational Operations

Generalized Projection

- **Example**

Consider the relation

EMPLOYEE (Ssn, Salary, Deduction, Years_service)

A report may be required to show

- Net Salary = Salary – Deduction,
- Bonus = 2000 * Years_service, and
- Tax = 0.25 * Salary.

Then a generalized projection combined with renaming may be used as follows:

- REPORT ← ρ(Ssn, Net_salary, Bonus, Tax)

$$(\pi_{\text{Ssn, Salary – Deduction, 2000 * Years\_service, 0.25 * Salary}}(\text{EMPLOYEE}))$$

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. These functions are used in simple statistical queries that summarize information from the database 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.

Additional Relational Operations

Use of the Functional operator $\mathcal{F}$

- $\mathcal{F}_{\text{MAX } \textit{Salary}}$ **(Employee)**
  retrieves the maximum salary value from the Employee relation

- $\mathcal{F}_{\text{MIN } \textit{Salary}}$ **(Employee)**
  retrieves the minimum Salary value from the Employee relation
- $\mathcal{F}_{\text{SUM } \textit{Salary}}$ **(Employee)**

  retrieves the sum of the Salary from the Employee relation
- ${}_{\text{DNO}}\,\mathcal{F}_{\textit{COUNT SSN, AVERAGE Salary}}$ **(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

- **Aggregate Functions and Grouping**

The aggregate function operation.

a. $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}(_{\text{Dno}}\ \Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE}))$.

b. $_{\text{Dno}}\ \Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

c. $\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

(a) R

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

(b)

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

(c)

Count_ssn	Average_salary
8	35125

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—that is, all EMPLOYEE e' directly supervised by e; all employees e'' directly supervised by each employee e'; all employees e''' directly supervised by each employee e''; and so on .
  - 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

Recursive Closure Operations

Example :

To retrieve all employees supervised by “James Borg”

$$BORG\_SSN \leftarrow \pi_{Ssn}(\sigma_{Fname=\text{'James'} \textbf{ AND } Lname=\text{'Borg'}}(EMPLOYEE))$$

$$SUPERVISION(Ssn1, Ssn2) \leftarrow \pi_{Ssn,Super\_ssn}(EMPLOYEE)$$

$$RESULT1(Ssn) \leftarrow \pi_{Ssn1}(SUPERVISION \bowtie_{Ssn2=Ssn} BORG\_SSN)$$

$$RESULT2(Ssn) \leftarrow \pi Ssn1(SUPERVISION \bowtie_{Ssn2=Ssn} RESULT1)$$

$$RESULT \leftarrow RESULT2 \cup RESULT1$$

Additional Relational Operations

Recursive Closure Operations

Example :

SUPERVISION

(Borg's Ssn is 888665555)

Ssn1 (Ssn)	Ssn2 (Super_ssn)
123456789	333445555
333445555	888665555
999887777	987654321
987654321	888665555
666884444	333445555
453453453	333445555
987987987	987654321
888665555	null

RESULT1

Ssn
333445555
987654321

(Supervised by Borg)

RESULT2

Ssn
123456789
999887777
666884444
453453453
987987987

(Supervised by Borg's subordinates)

RESULT

Ssn
123456789
999887777
666884444
453453453
987987987
333445555
987654321

(RESULT1 $\cup$ RESULT2)

Additional Relational Operations

- **The OUTER JOIN Operation**
  - In NATURAL JOIN 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.
  - ⟕ The left outer join operation keeps every tuple in the first or left relation R in ⟕ 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 ⟖ S.
  - A third operation, full outer join, denoted by ⟗ 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

$$TEMP \leftarrow (EMPLOYEE \; ⟕_{Ssn=Mgr\_ssn} DEPARTMENT)$$

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

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

Additional Relational Operations

- **OUTER UNION Operations**
  - The outer union operation was developed to take the union of tuples from two relations if the relations are not union compatible.
  - This operation will take the union of tuples in two relations R(X,Y) and S(X, Z) that are partially compatible, meaning that only some of their attributes, say X, are union compatible.
  - The attributes that are union compatible are represented only once in the result, and those attributes that are not union compatible from either relation are also kept in the result relation T(X,Y, Z).
  - **Example:**

An outer union can be applied to two relations whose schemas are STUDENT(Name, SSN, Department, Advisor) and

INSTRUCTOR(Name, SSN, Department, Rank).

Tuples from the two relations are matched based on having the same combination of values of the shared attributes—Name, SSN, Department. If a student is also an instructor, both Advisor and Rank will have a value; otherwise, one of these two attributes will be null.

The result relation STUDENT_OR_INSTRUCTOR will have the following attributes:

STUDENT_OR_INSTRUCTOR (Name, SSN, Department, Advisor, Rank)

EXAMPLES

Example : COMPANY

Schema diagram for the COMPANY relational database schema; the primary keys are underlined.

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Example 01

- Retrieve the name and address of all employees who work for the 'Research' department.

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

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

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

Example 02

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

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

Example 03

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

$$\text{DEPT5\_PROJS} \leftarrow \rho_{(\text{Pno})}(\pi_{\text{Pnumber}}(\sigma_{\text{Dnum}=5}(\text{PROJECT})))$$

$$\text{EMP\_PROJ} \leftarrow \rho_{(\text{Ssn, Pno})}(\pi_{\text{Essn, Pno}}(\text{WORKS\_ON}))$$

$$\text{RESULT\_EMP\_SSNS} \leftarrow \text{EMP\_PROJ} \div \text{DEPT5\_PROJS}$$

$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{RESULT\_EMP\_SSNS} * \text{EMPLOYEE})$$

Example 04

- Make a list of project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

$\text{SMITHS(Essn)} \leftarrow \pi_{\text{Ssn}} (\sigma_{\text{Lname='Smith'}}(\text{EMPLOYEE}))$

$\text{SMITH\_WORKER\_PROJS} \leftarrow \pi_{\text{Pno}}(\text{WORKS\_ON} * \text{SMITHS})$

$\text{MGRS} \leftarrow \pi_{\text{Lname, Dnumber}}(\text{EMPLOYEE} \bowtie_{\text{Ssn=Mgr\_ssn}} \text{DEPARTMENT})$

$\text{SMITH\_MANAGED\_DEPTS(Dnum)} \leftarrow \pi_{\text{Dnumber}} (\sigma_{\text{Lname='Smith'}}(\text{MGRS}))$

$\text{SMITH\_MGR\_PROJS(Pno)} \leftarrow \pi_{\text{Pnumber}}(\text{SMITH\_MANAGED\_DEPTS} * \text{PROJECT})$

$\text{RESULT} \leftarrow (\text{SMITH\_WORKER\_PROJS} \cup \text{SMITH\_MGR\_PROJS})$

Example 04

- Make a list of project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

$$\pi_{Pno}\ (\text{WORKS\_ON} \bowtie_{Essn=Ssn}(\pi_{Ssn}\ (\sigma_{Lname=\text{'Smith'}}(\text{EMPLOYEE}))) \cup \pi_{Pno}\ ((\pi_{Dnumber}\ (\sigma_{Lname=\text{'Smith'}}(\pi_{Lname,\ Dnumber}(\text{EMPLOYEE}))) \bowtie_{Ssn=Mgr\_ssn}\text{DEPARTMENT})) \bowtie_{Dnumber=Dnum}\text{PROJECT})$$

Example 05

- List the names of all employees with two or more dependents.

$$T1(\text{Ssn}, \text{No\_of\_dependents}) \leftarrow {}_{\text{Essn}}\,\Im\,{}_{\text{COUNT Dependent\_name}}(\text{DEPENDENT})$$

$$T2 \leftarrow \sigma_{\text{No\_of\_dependents}>2}(T1)$$

$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(T2 * \text{EMPLOYEE})$$

Example 06

- Retrieve the names of employees who have no dependents.

$$\text{ALL\_EMPS} \leftarrow \pi_{\text{Ssn}}(\text{EMPLOYEE})$$

$$\text{EMPS\_WITH\_DEPS}(\text{Ssn}) \leftarrow \pi_{\text{Essn}}(\text{DEPENDENT})$$

$$\text{EMPS\_WITHOUT\_DEPS} \leftarrow (\text{ALL\_EMPS} - \text{EMPS\_WITH\_DEPS})$$

$$\text{RESULT} \leftarrow \pi_{\text{Lname, Fname}}(\text{EMPS\_WITHOUT\_DEPS} * \text{EMPLOYEE})$$

$$\pi_{\text{Lname, Fname}}((\pi_{\text{Ssn}}(\text{EMPLOYEE}) - \rho_{\text{Ssn}}(\pi_{\text{Essn}}(\text{DEPENDENT}))) * \text{EMPLOYEE})$$

Example 07

- List the names of managers who have at least one dependent.

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

Reference

- *Chapter 6 - Fundamentals of Database Systems*

(6th Edition) By Remez Elmasri & Shamkant B. Navathe

Questions ???

Thank You