IN 1400 - Fundamentals of Databases and Database Design

RELATIONAL ALGEBRA

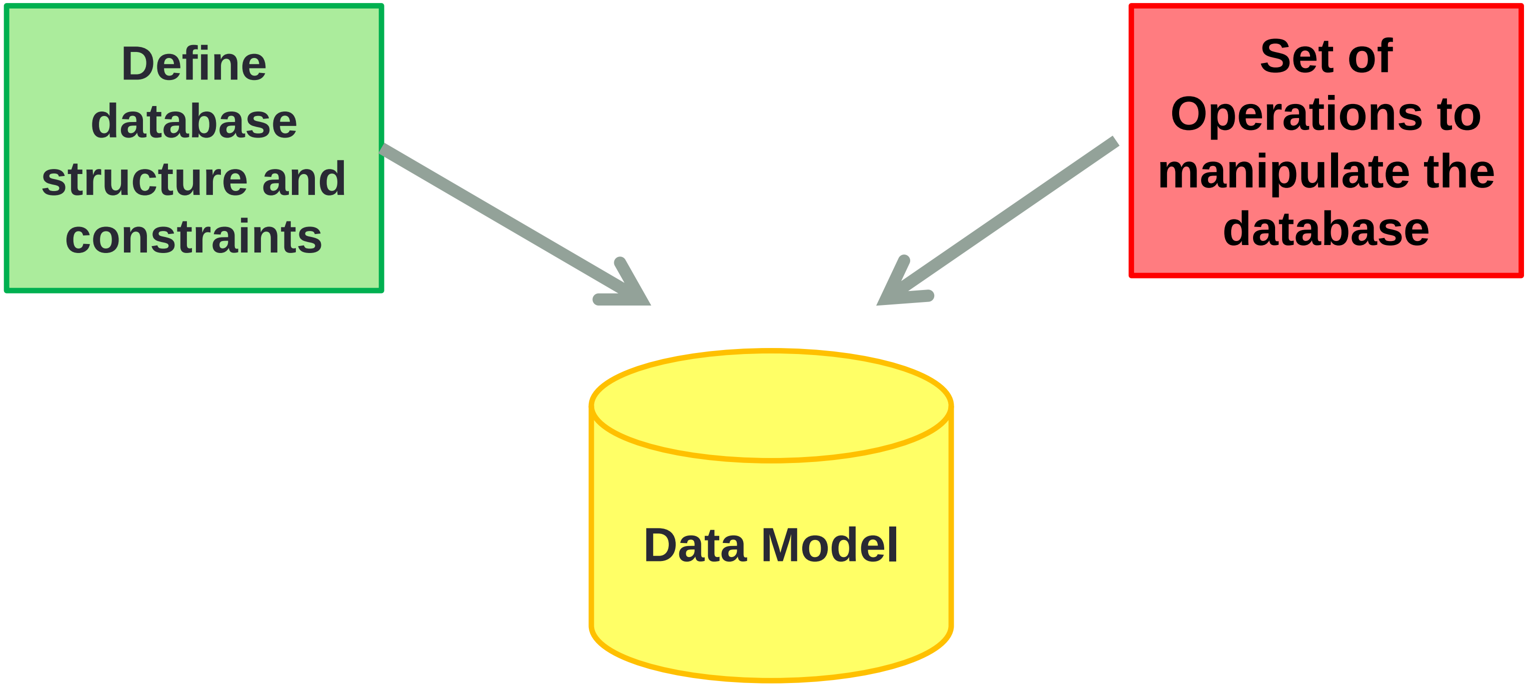
Week 10

Outline

- Unary Relational Operations
- Relational Algebra Operations From Set Theory
- Binary Relational Operations
- Additional Relational Operations
- Examples of Queries in Relational Algebra

Overview

- Formal language of Relational Model
  - Relational Algebra
  - Relational Calculus



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.

- 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**,
  - It 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).

Importance of Relational Algebra

- Formal foundation for relational model operations
- It is used as the basis for implementing and optimizing queries in RDBMS
- Its concepts are incorporated into SQL for RDBMS

Database State for COMPANY

EMPLOYEE

FNAME	MINIT	LNAME	<u>ENO</u>	BDATE	ADDRESS	SEX	SALARY	SUPERENO	DNO

DEPARTMENT

DNAME	<u>DNUMBER</u>	MGRENO	MGRSTARTDATE

DEPT_LOCATIONS

<u>DNUMBER</u>	<u>DLOCATION</u>

PROJECT

PNAME	<u>PNUMBER</u>	PLOCATION	DNUM

WORKS_ON

<u>EENO</u>	<u>PNO</u>	HOURS

DEPENDENT

<u>EENO</u>	<u>DEPENDENT_NAME</u>	SEX	BDATE	RELATIONSHIP

Unary Relational Operations

SELECT Operation

- SELECT operation is used to select a ***subset* of tuples** from a relation that satisfy a **selection condition**.
  - It is a filter that keeps only those tuples that satisfy a qualifying condition

Example: To select the EMPLOYEE tuples whose department number is four 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 SELECT operation σ is **commutative;** i.e.,

  $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(R)) = \sigma_{\text{<condition2>}}(\sigma_{\text{< condition1>}}(R))$

- A cascaded SELECT operation **may be applied in any order;** i.e.,

  $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(\sigma_{\text{<condition3>}}(R))$
  $= \sigma_{\text{<condition2>}}(\sigma_{\text{< condition3>}}(\sigma_{\text{< condition1>}}(R)))$

- A cascaded SELECT operation may be replaced by a single selection with a conjunction of all the conditions**;** i.e.,

  $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(\sigma_{\text{<condition3>}}(R))$
  $= \sigma_{\text{<condition1> AND < condition2> AND < condition3>}}(R)))$

Unary Relational Operations (cont.)

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:

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

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.

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

Unary Relational Operations (cont.)

PROJECT Operation Properties

- The number of tuples in the result of projection $\pi_{\text{<list>}}(R)$ is always less or equal to the number of tuples in R.
- If the list of attributes <u>includes a key of R</u>, then the number of tuples is equal to the number of tuples in R.
- If <list2> contains the attributes in <list1>, then

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

- Commutative does not hold in Project

$$\pi_{\text{<list1>}}(\pi_{\text{<list2>}}(R)) \neq \pi_{\text{<list2>}}(\pi_{\text{<list1>}}(P))$$

Example

- $\sigma_{\text{(DNO=4 AND Salary>25,000) OR (DNO=5 AND Salary>30,000)}}$ **(Employee)**
- $\pi_{\text{Lname, Fname, Salary}}$**(Employee)**

Unary Relational Operations (cont.)

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, or
  - We can apply one operation at a time and create **intermediate result relations**. Here, 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 (cont.)

- **Rename Operation (cont.)**

The rename operator is ρ (rho)

If the attributes of R are (A_1, A_2, …, A_n)

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

- $\rho_{S\,(B_1, B_2, \ldots, B_n)}$ **(R)** : Rename both Relation name and attribute names.
- ρ_{S}**(R)** : Rename Relation name (does not specify column names).
- $\rho_{(B_1, B_2, \ldots, B_n)}$ **(R)** : Rename relation with column names B_1, B_1, …..B_n (does not specify a new relation name).

Where **S** the new Relation name and **(B_1, B_2, …, B_n)** is the new attribute name list

Relational Algebra Operations From Set Theory

UNION Operation

- 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.

Example: To retrieve the Employee Number (ENO) 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:

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

$$\textbf{RESULT1} \leftarrow \pi_{\textbf{ENO}}\textbf{(DEP5\_EMPS)}$$

$$\textbf{RESULT2(ENO)} \leftarrow \pi_{\textbf{SUPERENO}}\textbf{(DEP5\_EMPS)}$$

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

- 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

- **Type Compatibility**
  - The operand relations $R_1(A_1, A_2, ..., A_n)$ and $R_2(B_1, B_2, ..., B_n)$ must have the <u>same number of attributes</u>, and the <u>domains of corresponding attributes must be compatible</u>; that is, **$dom(A_i) = dom(B_i)$** for i=1, 2, ..., n.
  - The resulting relation for $R_1 \cup R_2$, $R_1 \cap R_2$, or $R_1 - R_2$ has the <u>same attribute names as the *first* operand relation R_1</u> (by convention).

Union Example

Student

FN	LN
Peter	Parker
James	Anderson
Jessica	Barnes
Walter	Cooper

Instructor

FName	LName
Elizabeth	Johnson
Susan	Yeo
John	Mills
James	Anderson

STUDENT ∪ INSTRUCTOR

FN	LN
Peter	Parker
James	**Anderson**
Jessica	Barnes
Walter	Cooper
Elizabeth	Johnson
Susan	Yeo
John	Mills

Relational Algebra Operations From Set Theory (cont.)

- **INTERSECTION OPERATION**

The result of this operation, denoted by **R ∩ 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 - includes only those who are both students and instructors.

STUDENT ∩ INSTRUCTOR

FN	LN
James	Anderson

Relational Algebra Operations From Set Theory (cont.)

- **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: The names of students who are not instructors, and the names of instructors who are not students.

FN	**LN**
Peter	Parker
Jessica	Barnes
Walter	Cooper

STUDENT - INSTRUCTOR

FName	**LName**
Elizabeth	Johnson
Susan	Yeo
John	Mills

INSTRUCTOR - STUDENT

Relational Algebra Operations From Set Theory (cont.)

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

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

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

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

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

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

Relational Algebra Operations From Set Theory (cont.)

CARTESIAN PRODUCT (cross product or cross join) Operation

- This operation is used to combine tuples from two relations in a combinatorial fashion.
- In general, the result of $\mathbf{R(A_1, A_2, \ldots, A_n)}$ **x** $\mathbf{S(B_1, B_2, \ldots, B_m)}$ 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 x S | will have $n_R * n_S$ tuples.

- The two operands **do NOT have to be "type compatible**"

Example:

FEMALE_EMPS $\leftarrow \sigma_{SEX='F'}$**(EMPLOYEE)**
EMPNAMES $\leftarrow \pi_{FNAME, LNAME, SSN}$ **(FEMALE_EMPS)**

EMP_DEPENDENTS $\leftarrow$ **EMPNAMES x DEPENDENT**

Every tuple of EMPNAMES combined with every tuple from DEPENDENT
Meaningless

Example (Cont.)

FEMALE_EMPS $\leftarrow \sigma_{SEX='F'}$**(EMPLOYEE)**

EMPNAMES $\leftarrow \pi_{FNAME,\ LNAME,\ ENO}$ **(FEMALE_EMPS)**

EMP_DEPENDENTS $\leftarrow$ **EMPNAMES x DEPENDENT**

From DEPENDENT tuples

ACTUAL_DEPENDENTS $\leftarrow \sigma_{ENO=EENO}$ **(EMP_DEPENDENTS)**

RESULT $\leftarrow \pi_{FNAME,\ LNAME,\ DEPENDENT\_NAME}$ **(ACTUAL_DEPENDENTS)**

EMPLOYEE

FNAME	MINIT	LNAME	ENO	BDATE	ADDRESS	SEX	SALARY	SUPERENO	DNO

DEPENDENT

EENO	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

FEMALE_EMPS ← $\sigma_{SEX='F'}$ (EMPLOYEE)

FEMALE_EMPS	FNAME	MINT	LNAME	ENO	BDATE	ADDRESS	SEX	SALARY	SUPERENO	DNO
	Alice	J	Zebra	9910	1968-07-19	23, Castle St, TX	F	25000	8978	4
	Jennifer	S	Walsh	8978	1954-06-20	8/2, Houston, TX	F	43000	3321	4
	Joyce	A	Goyal	7771	1974-03-01	37. Main Rd, NY	F	25000	9993	5

EMPNAMES ← $\pi_{FNAME,\ LNAME,\ ENO}$ (FEMALE_EMPS)

EMPNAMES	FNAME	LNAME	ENO
	Alice	Zebra	9910
	Jennifer	Walsh	8978
	Joyce	Goyal	7771

EMP_DEPENDENTS	FNAME	LNAME	ENO	EENO	DEPENDENT_NAME	SEX	BDATE	….
	Alice	Zebra	9910	9993	Alex	M	1987-01-11	….
	Alice	Zebra	9910	9993	Thompson	M	1988-08-02	….
	Alice	Zebra	9910	9993	Joy	F	1958-11-21	….
	Alice	Zebra	9910	8978	Anandi	F	1990-12-24	….
	Alice	Zebra	9910	2221	Tejani	F	1967-09-09	….
	Alice	Zebra	9910	2221	Sammual	M	1995-08-11	….
	Alice	Zebra	9910	2221	Shinee	F	1998-11-30	….
	Jennifer	Walsh	8978	9993	Alex	M	1987-01-11	….
	Jennifer	Walsh	8978	9993	Thompson	M	1988-08-02	….
	Jennifer	Walsh	8978	9993	Joy	F	1958-11-21	….
	Jennifer	Walsh	8978	8978	Anandi	F	1990-12-24	….
	Jennifer	Walsh	8978	2221	Tejani	F	1967-09-09	….
	Jennifer	Walsh	8978	2221	Sammual	M	1995-08-11	….
	Jennifer	Walsh	8978	2221	Shinee	F	1998-11-30	….
	Joyce	Goyal	7771	9993	Alex	M	1987-01-11	….
	Joyce	Goyal	7771	9993	Thompson	M	1988-08-02	….
	Joyce	Goyal	7771	9993	Joy	F	1958-11-21	….
	Joyce	Goyal	7771	8978	Anandi	F	1990-12-24	….
	Joyce	Goyal	7771	2221	Tejani	F	1967-09-09	….
	Joyce	Goyal	7771	2221	Sammual	M	1995-08-11	….
	Joyce	Goyal	7771	2221	Shinee	F	1998-11-30	….

EMPNAMES x DEPENDENT

ACTUAL_DEPENDENTS ← $\sigma_{ENO=EENO}$ (EMP_DEPENDENTS)

ACTUAL_DEPENDENTS	FNAME	LNAME	ENO	EENO	DEPENDENT_NAME	SEX	BDATE	….
	Jennifer	Walsh	8978	8978	Anandi	F	1990-12-24	….

RESULT ← $\pi_{\text{FNAME, LNAME, DEPENDENT\_NAME}}$ (ACTUAL_DEPENDENTS)

RESULT	FNAME	LNAME	DEPENDENT_NAME
	Jennifer	Walsh	Anandi

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
  - It is denoted by a $\bowtie$
  - Only combination of tuples **satisfying the join condition** appear in the result.
  - The general form of a join operation on two relations

R(A_1, A_2, . . ., A_n) and **S(B_1, B_2, . . ., 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*.

Join Operation (Cont.)

- General Join Condition
- <condition> AND <condition> AND …. AND <condition>

- Each condition is in $A_i \ \theta \ B_j$
  - A_i is an attribute of R and B_j is an attribute of S. Θ (theta) is a comparison operator $\{=, <, \leq, >, \geq, \neq\}$

Join Operation (cont.)

Example: Suppose that we want to retrieve the names of the managers of each department.

To get the manager's name, we need to **combine** each **DEPARTMENT** tuple with the **EMPLOYEE** tuple whose ***ENO*** *value* ***matches*** *the* ***MGRENO*** *value in the department tuple.*

We do this by using the join operation.

$$\texttt{DEPT\_MGR} \leftarrow \texttt{DEPARTMENT} \bowtie_{\texttt{MGRENO=ENO}} \texttt{EMPLOYEE}$$

DEPT_MGR

DNAME	DNO	MGRENO	…	FNAME	MINIT	LNAME	ENO	…
Research	5	3334	…	Franklin	T	Wong	3334	….
Admin	4	9876	….	Jennifer	S	Borg	9876	…
HQ	1	8886	….	James	E	Wales	8886	…

Binary Relational Operations (cont.)

EQUIJOIN Operation

- The most common use of join involves join conditions with equality comparisons only.
- In EQUIJOIN, the <u>only comparison operator used is =.</u>

- 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.

$$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRENO=ENO}} \text{EMPLOYEE}$$

EQUIJOIN Vs. Natural Join

Values of MGRENO and ENO are **identical**

$$\text{DEPT\_MGR} \leftarrow \text{DEPARTMENT} \bowtie_{\text{MGRENO=ENO}} \text{EMPLOYEE}$$

DEPT_MGR

DNAME	DNO	MGRENO	…	FNAME	MINIT	LNAME	ENO	…
Research	5	3334	…	Franklin	T	Wong	3334	….
Admin	4	9876	….	Jennifer	S	Borg	9876	…
HQ	1	8886	….	James	E	Wales	8886	…

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.

Binary Relational Operations (cont.)

NATURAL JOIN Operation

- 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.

- First rename DNUMBER attribute in DEPARTMENT to DNUM. That is, it has the same name as the DNUM in PROJECT

$$\textbf{PROJ\_DEPT} \leftarrow$$

$$\textbf{PROJECT} * \rho_{\textbf{(DNAME, DNUM, MGRENO, MGRSTARTDATE)}}\textbf{(DEPARTMENT)}$$

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

DEPARTMENT

DNAME	DNUMBER	MGRENO	MGRSTARTDATE

Natural Join Operations (cont.)

Example: To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write:

DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS

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

DEPT_LOCATIONS

DNUMBER	DLOCATION

DEPARTMENT

DNAME	DNUMBER	MGRENO	MGRSTARTDATE

Additional Relational Operations (cont.)

The OUTER JOIN Operation

- In **Natural Join**
  - tuples without a *matching* (or *related*) tuple are eliminated from the join result.
  - Also, tuples with null in the join attributes are eliminated.
  - This leads to loss of information.
  - It is called **Inner Join**

- In **Outer Joins**
  - **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.

Additional Relational Operations (cont.)

- **Left outer join** operation
  - Keeps every tuple in the *first* or *left* relation R in R ⟕ 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.

- Similarly, **Right outer join**,
  - keeps every tuple in the *second* or right relation S in the result of R ⟖ S.

- **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 (cont.)

Result of **Left Outer Join**

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

Complete Set of Relational Operations

- The set of operations including **select σ, project π , union ∪, 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.

- For example:

$$\mathbf{R \cap S = (R \cup S) - ((R - S) \cup (S - R))}$$

$$R \bowtie_{\text{<join condition>}} S = \sigma_{\text{<join condition>}} \mathbf{(R \text{ X } S)}$$

Binary Relational Operations (cont.)

DIVISION Operation

- The division operation is applied to two relations R(Z) ÷ S(X), where X subset Z.

- Let Y = Z - X (and hence Z = X ∪ Y);
  - That is, let Y be the set of attributes of R that are not attributes of S.

- The result of DIVISION is a relation T(Y) that includes a tuple t if tuples t_R appear in R with t_R [Y] = t, and with t_R [X] = t_s *for every tuple* t_s in S.

Division Example

$T \leftarrow R \div S$

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.

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

Division Example

- Query: Retrieve the names of employees who work on all the projects that "Thilak Perera" works on.

1. Retrieve the list of project numbers that 'Thilak Perera' works on.
   - THILAK $\leftarrow \sigma_{\text{FNAME='THILAK' AND LNAME='PERERA'}}$ (EMPLOYEE)
   - THILAK_PNOS $\leftarrow \pi_{\text{PNO}}$ (WORKS_ON $\bowtie_{\text{EENO=ENO}}$ THILAK)

2. Retrieve the ENO and their project number from WORKS_ON relation
   - ENO_PNOS $\leftarrow \pi_{\text{EENO, PNO}}$ (WORKS_ON)

Example (cont.)

$ENOS \leftarrow ENO\_PNOS \div THILAK\_PNOS$

ENO_PNOS

EENO	PNO
123456789	1
123456789	2
666884444	3
4535453453	1
4535453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10

THILAK_PNOS

PNO
1
2

ENOS

ENO
123456789
4535453453

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 (cont.)

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

$\mathcal{F}_{\text{MAX Salary}}$ **(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

${}_{\text{DNO}}\,\mathcal{F}_{\text{COUNT ENO, AVERAGE Salary}}$ **(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]

Example

- $_{DNO}\ \mathcal{F}_{COUNT\ ENO,\ AVERAGE\ SALARY}$(EMPLOYEE)

DNO	COUNT_ENO	AVERAGE_SALARY
5	4	33250
4	3	31000
1	1	55000

- $\rho_{R\ (DNO,\ NO\_OF\_EMPLOYEES,\ AVERAGE\_SAL)}$ ($_{DNO}\ \mathcal{F}_{COUNT\ ENO,\ AVERAGE\ SALARY}$(EMPLOYEE))

R	DNO	NO_OF_EMPLOYEES	AVERAGE_SAL
	5	4	33250
	4	3	31000
	1	1	55000

- $\mathcal{F}_{COUNT\ ENO,\ AVERAGE\ SALARY}$(EMPLOYEE)

COUNT_ENO	AVERAGE_SALARY
8	35125

Examples of Queries in Relational Algebra

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

  RESEARCH_DEPT $\leftarrow \sigma_{\text{DNAME='Research'}}$ (DEPARTMENT)

  RESEARCH_EMPS $\leftarrow$ (RESEARCH_DEPT $\bowtie_{\text{DNUMBER= DNOEMPLOYEE}}$ EMPLOYEE)

  RESULT $\leftarrow \pi_{\text{FNAME, LNAME, ADDRESS}}$ (RESEARCH_EMPS)

- **Retrieve the names of employees who have no dependents.**

  ALL_EMPS $\leftarrow \pi_{\text{ENO}}$(EMPLOYEE)

  EMPS_WITH_DEPS(ENO) $\leftarrow \pi_{\text{EENO}}$(DEPENDENT)

  EMPS_WITHOUT_DEPS $\leftarrow$ (ALL_EMPS - EMPS_WITH_DEPS)

  RESULT $\leftarrow \pi_{\text{LNAME, FNAME}}$ (EMPS_WITHOUT_DEPS * EMPLOYEE)