DSAA 5012
Advanced Database Management for Data Science

LECTURE 5
RELATIONAL ALGEBRA（Cont.）

RELATIONAL ALGEBRA: OUTLINE

Relational Algebra

Basic Operations
- Selection
- Projection
- Union
- Set difference
- Cartesian product

Additional Operations
- Intersection
- Join
- Assignment
- Rename
- Div

RELATIONAL QUERY LANGUAGES

- Two mathematical query languages form the basis for "real" relational query languages (e.g., SQL) and for implementation.

Our focus

Relational Algebra — Procedural (*step-by-step*). Need to describe how to compute a query result.

Relational Calculus — Non-procedural (*declarative*). Only need to describe what query result is wanted, not how to compute it.

☞ **Relational algebra is very useful for representing and optimizing query execution plans.**

Understanding relational algebra is the key to understanding SQL and how it is processed!

RELATIONAL ALGEBRA

- The relational algebra is an algebra whose
  - **operands** are either relations or variables that represent relations.
  - **operations** perform common, basic manipulations of relations.

☞ A relational algebra expression is evaluated from the inside-out.

Closure Property

- Relational algebra is closed with respect to the relational model.

☞ Each operation manipulates one or more relations and returns a relation as its result.

Due to the closure property, operations can be *composed*!

RELATIONAL ALGEBRA: OUTLINE

✓ Relational Algebra

✓ Basic Operations
- Selection
- Projection
- Union
- Set difference
- Rename
- Cartesian product

➡ **Additional Operations**
- **Intersection**
- **Join**
- **Assignment**
- **Div**

INTERSECTION: ∩

Query: Find tuples that appear in both $Plane_1$ and $Plane_2$.

$Plane_1$

company	model
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B747
Boeing	B777

∩

$Plane_2$

company	model
Comac	C929
Comac	C939
Boeing	B747
Boeing	B777

=

company	model
Boeing	B747
Boeing	B777

$Plane_2$

company	model
Comac	C929
Comac	C939
Boeing	B747
Boeing	B777

∩

$Plane_1$

company	model
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B747
Boeing	B777

= ?

company	model

INTERSECTION (Cont.)

- Intersection is not a primitive operation

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

Compute all tuples belonging to R or S

Remove the ones that belong only to R

Remove the ones that belong only to S

Also: R□S = R□ (R□S)

OUTER JOIN

- An extension of the natural join operation that avoids loss of information.

- Computes the natural join and then adds tuples from one relation that do not have matching tuples in the other relation to the result of the join.

- Uses null values to fill in missing information.
  - Recall that null signifies that the value is unknown or does not exist.

☞ **All comparisons involving null are false.**

OUTER JOIN (cont'd)

Loan

loan Number	amount	branch Name
L-170	30000	Central
L-260	170000	Tsimshatsui
L-230	40000	Central

$\bowtie$

Borrower

client Name	loan Number
Pat Lee	L-170
Mary Kwan	L-230
Ted Hayes	L-155

=

loan Number	amount	branch Name	client Name
L-170	30000	Central	Pat Lee
L-230	40000	Central	Mary Kwan

- Natural join returns only the tuples that match on the join attributes (the "good tuples").
- The fact that
  - loan L-260 has no borrower is not explicit in the result.
  - customer Ted Hayes holds a non-existent loan L-155 with no amount and no branch is also not explicit.

LEFT OUTER JOIN: $⟕$

Adds to the natural join all tuples in the left relation (Loan) that did not match with any tuple in the right relation (Borrower) and fills in null for the missing information.

Loan

loan Number	amount	branch Name
L-170	30000	Central
L-260	170000	Tsimshatsui
L-230	40000	Central

$⟕$

Borrower

client Name	loan Number
Pat Lee	L-170
Mary Kwan	L-230
Ted Hayes	L-155

=

loan Number	amount	branch Name	client Name
L-170	30000	Central	Pat Lee
L-230	40000	Central	Mary Kwan
L-260	170000	Tsimshatsui	null

☞ The result now shows that loan L-260 has no borrower.

RIGHT OUTER JOIN: ⟖

Adds to the natural join all tuples in the right relation (Borrower) that did not match with any tuple in the left relation (Loan) and fills in null for the missing information.

Loan

loan Number	amount	branch Name
L-170	30000	Central
L-260	170000	Tsimshatsui
L-230	40000	Central

⟖

Borrower

client Name	loan Number
Pat Lee	L-170
Mary Kwan	L-230
Ted Hayes	L-155

=

loan Number	amount	branch Name	client Name
L-170	30000	Central	Pat Lee
L-230	40000	Central	Mary Kwan
L-155	null	null	Ted Hayes

☞ The result now shows that loan L-155 has no amount and no branch.

FULL OUTER JOIN: ⟗

Adds to the natural join all tuples in both relations that did not match with any tuples in the other relation and fills in null for missing information.

Loan

loan Number	amount	branch Name
L-170	30000	Central
L-260	170000	Tsimshatsui
L-230	40000	Central

⟗

Borrower

client Name	loan Number
Pat Lee	L-170
Mary Kwan	L-230
Ted Hayes	L-155

=

loan Number	amount	branch Name	client Name
L-170	30000	Central	Pat Lee
L-230	40000	Central	Mary Kwan
L-260	170000	Tsimshatsui	null
L-155	null	null	Ted Hayes

☞ The result now shows both that

- loan L-260 has no borrower.
- loan L-155 has no amount and no branch.

ASSIGNMENT: $\leftarrow$

- Works like assignment in programming languages.
- The relation variable assigned to can be used in subsequent expressions.
- Allows a query to be written as a sequential program consisting of a series of assignments followed by an expression whose value is the result of the query.
- Useful for expressing complex queries.

RENAMING: ρ

- Assigns a name to, or renames the attributes in, a relational-algebra expression.

$\rho_x(E)$	assigns name x to the result of E
$\rho_{x(A1,\ A2,\ \ldots,\ An)}(E)$	assigns name x to the result of E *and* renames the attributes of E as $A_1, A_2, \ldots, A_n$

☞ **Renaming is necessary when taking the Cartesian product of a table with itself.**

RENAMING: ρ

- If attributes or relations have the same name it may be convenient to rename one

$$\rho(R'(N_1 \text{ -> } N'_1, N_n \text{ -> } N'_n), R)$$

- The new relation R′ has the same instance as R, but its schema has attribute N'_i instead of attribute N_i
- **Example:** ρ(Staff(Name -> Family_Name, Salary -> Gross_salary), Employee)

- Necessary if we need to perform a cartesian product or join of a table with itself

Employee

Name	Salary	Emp_No
Clark	150000	1006
Gates	5000000	1005
Jones	50000	1001
Peters	45000	1002
Phillips	25000	1004
Rowe	35000	1003
Warnock	500000	1007

Staff

Family_Name	Gross_Salary	Emp_No
Clark	150000	1006
Gates	5000000	1005
Jones	50000	1001
Peters	45000	1002
Phillips	25000	1004
Rowe	35000	1003
Warnock	500000	1007

DIVISION: /

Let A have two fields x and y

Let B have one field y

A/B contains all x tuples, such that for **every** y tuple in B there is a xy tuple in A

A

x	y
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

/

B

y
p2
p4

=

x
s1
s4

A/B

DIVISION: / (Cont.)

- Compute all possible combinations of the first column of A and B.
- Then remove those rows that exist in A
- Keep only the first column of the result. These are the ***disqualified*** values
  - $\pi_x((\pi_x(A) \times B) - A)$
- A/B is the first column of A **except** the disqualified values
  - $A/B = \pi_x(A) - \pi_x((\pi_x(A) \times B) - A)$

DIVISION: / (Cont.)

$\pi_x(A) = \pi_x($

x	y
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

$) =$

x
s1
s2
s3
s4

$\pi_x(A) \times B =$

x
s1
s2
s3
s4

$\times$

y
p2
p4

$=$

x	y
s1	p2
s1	p4
s2	p2
s2	p4
s3	p2
s3	p4
s4	p2
s4	p4

DIVISION: / (Cont.)

$$(\pi_x(A) \times B) - A =$$

x	y
s1	p2
s1	p4
s2	p2
s2	p4
s3	p2
s3	p4
s4	p2
s4	p4

−

x	y
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

=

x	y
s2	p4
s3	p4

$$\pi_x(A) - \pi_x(\pi_x(A) \times B) - A) =$$

x
s1
s2
s3
s4

−

x
s2
s3

=

x
S1
S4

DIVISION EXAMPLE

Find the Employment numbers of the pilots who can fly **all** MD planes

$Can\_Fly\ /\ \pi_{Model\_No}(\sigma_{Maker='MD'}Plane)$

Emp_No	Model_No
1001	B727
1001	B747
1001	DC10
1002	A320
1002	A340
1002	B757
1002	DC9
1003	A310
1003	DC9
1003	DC10

Maker	Model_No
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

Emp_No
1003

RELATIONAL ALGEBRA: SUMMARY

- Defines a set of algebraic operations that operate on relations and output relations as their result.
- The operations can be combined to express queries.
- The operations can be divided into:
  - basic operations.
  - additional operations that either
    - can be expressed in terms of the basic operations or
    - add further expressive power to the relational algebra.

COMP 3311: SYLLABUS

- ✓ Introduction
- ✓ Entity-Relationship (E-R) Model and Database Design
- ✓ Relational Algebra
- ➡ **Structured Query Language (SQL)**
- Relational Database Design
- Storage and File Structure
- Indexing
- Query Processing
- Query Optimization
- Transactions
- Concurrency Control
- Recovery System
- NoSQL Databases