COMP 3311
DATABASE MANAGEMENT SYSTEMS

LECTURE 6
RELATIONAL ALGEBRA

RELATIONAL ALGEBRA: OUTLINE

Relational Algebra

Basic Operations

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

Additional Operations

- Intersection
- Join
- Assignment
- Rename

EXAMPLE RELATIONAL SCHEMA AND DATABASE

Sailor(sailorId, sName, rating, age)

Boat(boatId, bName, color)

Reserves(*sailorId, boatId*, rDate)

Attribute names in italics are foreign key attributes.

Sailor

sailorId	sName	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	35
85	Art	3	25
95	Bob	3	63
99	Chris	10	30

11 tuples

Reserves

sailorId	*boatId*	rDate
22	101	10/10/17
22	102	10/10/17
22	103	08/10/17
22	104	07/10/17
31	102	10/11/17
31	103	06/11/17
31	104	12/11/17
64	101	05/09/17
64	102	08/09/17
74	103	08/09/17
99	104	08/08/17

11 tuples

Boat

boatId	bName	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red
105	Serenity	Cyan

5 tuples

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: BASIC OPERATIONS

Operation	Symbol	Action
Selection	σ	Selects rows in a table that satisfy a predicate
Projection	π	Removes unwanted columns from a table
Union	$\cup$	Finds rows that belong to either table 1 or table 2
Set difference	$-$	Finds rows that are in table 1, but are not in table 2
Cartesian product	$\times$	Allows the rows in two tables to be combined

Additional operations *(not essential, but very useful)*:

Intersection	$\cap$	Finds tuples that appear in both table 1 and in table 2
Join	$\bowtie$	Cartesian product followed by a selection
Assignment	$\leftarrow$	Assigns a result to a temporary variable
Rename	ρ	Allows a table and/or its columns to be renamed

SELECTION: $\sigma_C(R)$

- Selects tuples (rows) that satisfy a *selection condition C*.
- The schema of the result is identical to the schema of the (only) input relation.
- A condition C has the form: **term** *op* **term** where
  - **term** is an attribute name or a constant
  - *op* is a comparison operator such as =, ≠, <, ≤, >, ≥.
- Conditions can be composed or negated using Boolean operators.

$C_1 \wedge C_2$	where C_1 and C_2 are conditions and $\wedge$ means AND
$C_1 \vee C_2$	where C_1 and C_2 are conditions and $\vee$ means OR
$\neg C$	where $\neg$ means NOT

SELECTION: EXAMPLE

Query: Find tuples where the company is Boeing.

Plane

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

$\sigma_{\text{company='Boeing'}}(\text{Plane})$ =

company	model
Boeing	B747
Boeing	B777

Query: Find tuples where the company is Boeing, or the model is A330.

Plane

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

$\sigma_{\text{company='Boeing'} \vee \text{model='A330'}}(\text{Plane})$ = ?

company	model

PROJECTION: $\pi_L(R)$

- Keeps only the attributes (columns) in a *projection list L*.

  ☞ The schema of the result contains the same attributes as in the projection list L, with the same names that they had in the (only) input relation.

- The projection operator *eliminates duplicate tuples*. **Why?**

Query: Find the companies that make planes.

Plane

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

$\pi_{company}(Plane)$ =

company
Airbus
Boeing

COMPOSITION OF OPERATIONS

- Since relational algebra operations are closed, the result of one relational algebra operation can be the input for another relational algebra operation (i.e., operations can be composed).

☞ **The result of a relational algebra operation must be a relation.**

Query: Find only those models made by Boeing.

Plane

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

$\boxed{\pi_{model}(\sigma_{company='Boeing'}(Plane))}$ =

model
B747
B777

Is this a correct solution? $\boxed{\sigma_{company='Boeing'}(\pi_{model}(Plane))}$

SET OPERATIONS

- The set operations are:

  $\cup$ union

  $-$ set difference

  $\cap$ intersection (not basic; can be expressed using only set-difference, i.e., $r \cap s = r - (r - s)$)

- These operations take two input relations, which must be union-compatible, which means that
  - the relations have the same number of attributes.
  - corresponding attributes have the same type.

- The output is a single relation (without duplicates).

UNION: ∪

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

$Plane_1$

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
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B747
Boeing	B777
Comac	C929
Comac	C939

Appears only once in the result since set operations eliminate duplicates.

SET DIFFERENCE: –

Query: Find tuples that appear in $Plane_1$, but not in $Plane_2$.

$Plane_1$

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
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340

Removed from the result since they appear in both relations.

$Plane_2$

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

CARTESIAN PRODUCT: ×

- Cartesian product combines each row of one table with every row of another table.

- CanFly × Plane ⇒ **72 tuples**!!!
  (9 × 8)

CanFly

empNo	model
1001	B747
1001	B777
1001	A310
1002	A320
1002	A340
1002	B777
1002	C929
1003	A310
1003	C939

×

Plane

company	model
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B747
Boeing	B777
Comac	C929
Comac	C939

=

empNo	model	company	model
1001	B747	Airbus	A310
1001	B747	Airbus	A320
1001	B747	Airbus	A330
1001	B747	Airbus	A340
1001	B747	Boeing	B747
1001	B747	Boeing	B777
1001	B747	Comac	C929
1001	B747	Comac	C939
1001	B777	Airbus	A310
1001	B777	Airbus	A320
1001	B777	Airbus	A330
1001	B777	Airbus	A340
1001	B777	Boeing	B747
1001	B777	Boeing	B777
1001	B777	Comac	C929
1001	B777	Comac	C939
1001	A310	Airbus	A310
1001	A310	Airbus	A320
⋮	⋮	⋮	⋮

INTERSECTION: $\cap$

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

$Plane_1$ $\cap$ $Plane_2$ =

$Plane_1$

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

$\cap$

$Plane_2$

company	model
Comac	C929
Comac	C939
Boeing	B747
Boeing	B777

=

company	model
Boeing	B747
Boeing	B777

$Plane_2$ $\cap$ $Plane_1$ = ?

$Plane_2$

company	model
Comac	C929
Comac	C939
Boeing	B747
Boeing	B777

$\cap$

$Plane_1$

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

= ?

company	model

JOIN: ⋈

- Generating all possible tuple combinations of two relations is usually not meaningful.

  Example: For the relations CanFly and Plane, combining each CanFly and Plane tuple having a matching model value is more meaningful than CanFly $\times$ Plane.

- Join is a Cartesian product followed by a selection:

  $R_1 \bowtie_c R_2 = \sigma_c(R_1 \times R_2)$ *or* $R_1 \text{ JOIN}_c R_2 = \sigma_c(R_1 \times R_2)$

- Types of joins:

  natural join — Combines two relations on the equality of the attribute values with the same names.

  θ-join — Allows arbitrary conditions in the selection.

  equijoin — All conditions are equality.

  ☞ **Both equijoin and natural join project the result on <u>only one</u> of the join attributes.**

CanFly

empNo	model
1001	B747
1001	B777
1001	A310
1002	A320
1002	A340
1002	B777
1002	C929
1003	A310
1003	C939

Plane

company	model
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B747
Boeing	B777
Comac	C929
Comac	C939

JOIN: NATURAL JOIN

CanFly $\bowtie_n$ Plane ⇔ CanFly $\bowtie$ Plane

CanFly JOIN$_n$ Plane ⇔ CanFly JOIN Plane

CanFly JOIN$_{model}$ Plane

CanFly JOIN$_{CanFly.model=Plane.model}$ Plane

n ⟹ look for attributes with common names in the two relations.

CanFly

empNo	model
1001	B747
1001	B777
1001	A310
1002	A320
1002	A340
1002	B777
1002	C929
1003	A310
1003	C939

$\bowtie_n$

Plane

company	model
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B747
Boeing	B777
Comac	C929
Comac	C939

=

empNo	model	company
1001	B747	Boeing
1001	B777	Boeing
1001	A310	Airbus
1002	A320	Airbus
1002	A340	Airbus
1002	B777	Boeing
1002	C929	Comac
1003	A310	Airbus
1003	C939	Comac

Cartesian product ⟹ 72 tuples; join ⟹ 9 tuples.

JOIN: θ-JOIN

- If we join this table with itself (*self-join*) using the condition:

  c = Flight1.destination=Flight2.origin ∧ Flight1.arrivalTime<Flight2.deptartureTime

What should we get?

Flight1

flight#	origin	destination	departure Time	arrival Time
334	HKG	PVG	12:00	14:14
335	PVG	HKG	15:00	17:14
336	HKG	PVG	18:00	20:14
337	PVG	HKG	20:30	23:53
394	PEK	PVG	19:00	21:30
395	PVG	PEK	21:00	23:43

$\bowtie_c$

Flight2

flight#	origin	destination	departure Time	arrival Time
334	HKG	PVG	12:00	14:14
335	PVG	HKG	15:00	17:14
336	HKG	PVG	18:00	20:14
337	PVG	HKG	20:30	23:53
394	PEK	PVG	19:00	21:30
395	PVG	PEK	21:00	23:43

JOIN: θ-JOIN (CONT'D)

$$\text{Flight1} \bowtie_{\text{Flight1.destination=Flight2.origin} \land \text{Flight1.arrivalTime<Flight2.departureTime}} \text{Flight2}$$

Flight1. Flight#	Flight1. Origin	Flight1. Destination	Flight1. Departure Time	Flight1. Arrival Time	Flight2. Flight#	Flight2. Origin	Flight2. Destination	Flight2. Departure Time	Flight2. Arrival Time
334	HKG	PVG	12:00	14:14	335	PVG	HKG	15:00	17:14
335	PVG	HKG	15:00	17:14	336	HKG	PVG	18:00	20:14
336	HKG	PVG	18:00	20:14	337	PVG	HKG	20:30	23:53
334	HKG	PVG	12:00	14:14	337	PVG	HKG	20:30	23:53
336	HKG	PVG	18:00	20:14	395	PVG	PEK	21:00	23:43
334	HKG	PVG	12:00	14:14	395	PVG	PEK	21:00	23:43

What happens if we add the condition: … ∧ Flight1.origin<>Flight2.destination?

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.**

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.