COMP 33II 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(sailorld, sName, rating, age)

Boat(boatId, bName, color)

Reserves(<u>sailorId</u>, <u>boatId</u>, <u>rDate</u>)

Attribute names in italics are foreign key attributes.

Sailor

<u>sailorld</u>	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

<u>sailorId</u>	<u>boatld</u>	<u>rDate</u>
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

Boat

<u>boatld</u>	bName	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red
105	Serenity	Cyan

5 tuples

11 tuples



RELATIONAL QUERY LANGUAGES

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

Relational Procedural (*step-by-step*).

Algebra

Need to describe how to compute a query result.

Relational

Non-procedural (*declarative*).

Calculus

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 <u>closed</u> with respect to the relational model.

relations and returns a relation as its result.

Due to the closure property, operations can be <u>composed!</u>



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	U	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	×	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	M	Cartesian product followed by a selection
Assignment	\leftarrow	Assigns a result to a temporary variable
Rename	n	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 =, \neq , <, ≤, >, ≥.
- 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

¬ C where ¬ 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



company	model
Boeing	B747
Boeing	B777

Plane

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

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

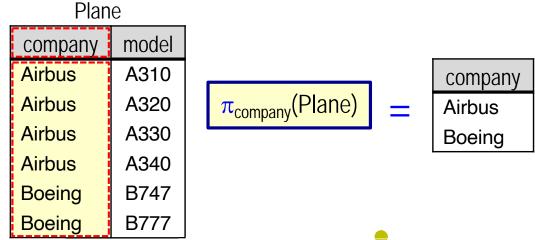


company	model

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.



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.

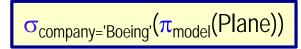
Query: Find only those models made by Boeing.

D	la	n△
Г	ıa	HE

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

$$\frac{\pi_{\text{model}}(\sigma_{\text{company='Boeing'}}(\text{Plane}))}{\text{B747}} = \frac{\text{model}}{\text{B777}}$$

Is this a correct solution?



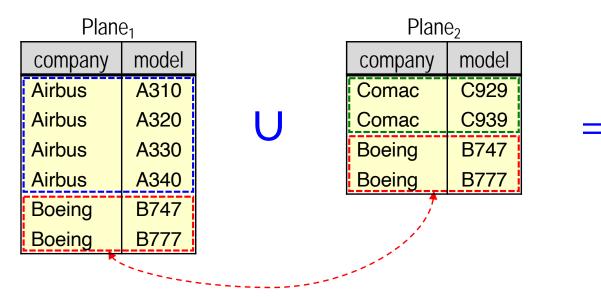


SET OPERATIONS

- The set operations are:
 - **U** union
 - set difference
 - intersection (not basic; can be expressed using only setdifference, i.e., $r \cap s = r - (r - s)$)
- These operations take two input relations, which must be <u>union-compatible</u>, 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: U

Query: Find tuples that appear in Plane₁, Plane₂ or both.

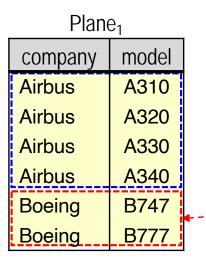


company	model
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B747
Boeing	B777
Comac	C929
Comac	C939

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

SET DIFFERENCE: -

Query: Find tuples that appear in Plane₁, but not in Plane₂.



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.

company	model
Comac	C929
Comac	C939
Boeing	B747
Boeing	B777



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



company	model

model

A310

A320

A320

company

Airbus

Airbus

CARTESIAN PRODUCT: X

- Cartesian product combines each row of one table with every row of another table.
- CanFly \times Plane \Rightarrow **72 tuples**!!!

CanFly

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



Plane

modei
A310
A320
A330
A340
B747
B777
C929
C939

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

empNo

1001

1001

1001

model

B747

B747



Airbus

A310

INTERSECTION: \(\)

Query: Find tuples that appear in both Plane₁ and Plane₂.

Plane₁

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

Plane₂

company	model
Comac	C929
Comac	C939
Boeing	B747
Boeing	B777

company	model
Boeing	B747
Boeing	B777

Plane₂

company	model
Comac	C929
Comac	C939
Boeing	B747
Boeing	B777

Plane₁

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

= ?

company	model

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 × Plane.

Join is a Cartesian product followed by a selection:

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

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

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 only one 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 \Leftrightarrow CanFly \bowtie 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			

CapEly



model
A310
A320
A330
A340
B747
B777
C929
C939

Plane

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

Cartesian product \Rightarrow 72 tuples; join \Rightarrow 9 tuples.





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

c = Flight1.destination=Flight2.origin \(\sim \) 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

M

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: 0-JOIN (CONTD)

Flight1 ⋈ Flight1.destination=Flight2.origin ∧ Flight1.arrivalTime<Flight2.departureTime 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 (CONTD)

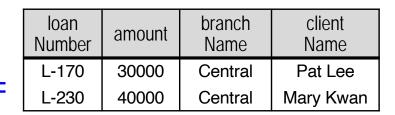
Loan

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



Borrower

loan Number
L-170
L-230
L-155



- 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: D

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

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

RIGHT OUTER JOIN: K

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			_	Borro	ver					
loan Number	amount	branch Name		client Name	loan Number		loan Number	amount	branch Name	client Name
L-170	30000	Central		Pat Lee	L-170		L-170	30000	Central	Pat Lee
L-260	170000	Tsimshatsui	$ \mathbf{X} $	Mary Kwan	L-230	=	L-230	40000	Central	Mary Kwan
L-230	40000	Central		Ted Hayes	L-155	>	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					Borrov						
	loan Number	amount	branch Name		client Name	loan Number		loan Number	amount	branch Name	client Name
	L-170	30000	Central		Pat Lee	L-170		L-170	30000	Central	Pat Lee
	L-260	170000	Tsimshatsui -		Mary Kwan	L-230	=	L-230	40000	Central	Mary Kwan
	L-230	40000	Central		Ted Hayes	L-155	>	L-260	170000	Tsimshatsui	null
_				·			1.77	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: ←

- 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: P

 Assigns a name to, or renames the attributes in, a relationalalgebra expression.

 $\rho_{x}(E)$ assigns name x to the result of E assigns name x to the result of E assigns name x to the result of E and renames the attributes of E as $A_{1}, A_{2}, ..., A_{n}$

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.