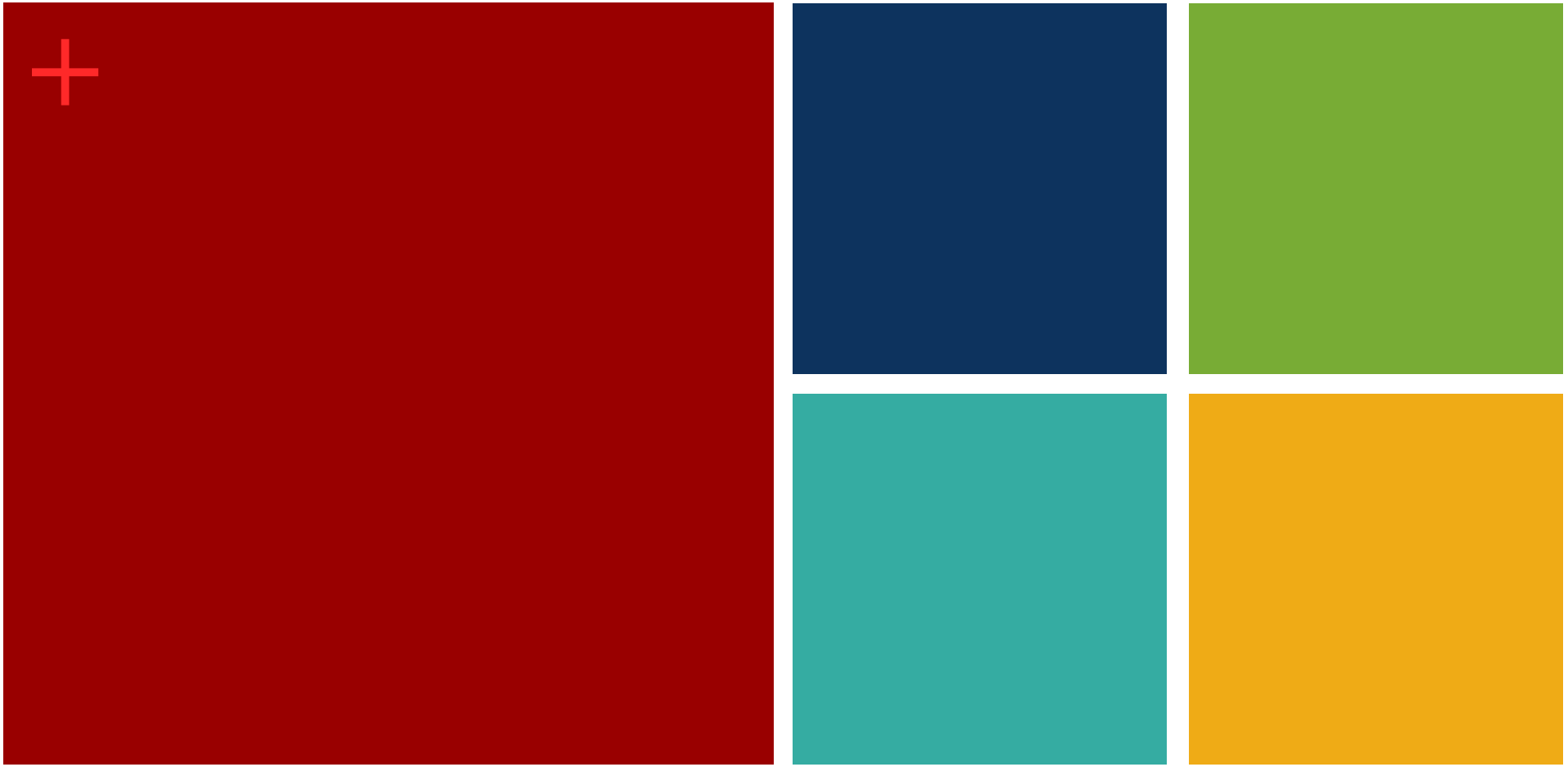


COMP3311 Database Management Systems

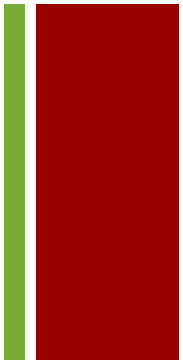
Spring 2022

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Relational Algebra

Prof Xiaofang Zhou

+ Relational Query Languages



- Query languages (QL): allow retrieval of data from a database
- Relational model supports simple, powerful QLs:
  - Formal foundation based on logic
  - Allows for optimization
- Query Languages ≠ programming languages!
  - QLs not expected to be “Turing complete”
  - QLs not intended to be used for complex calculations
  - QLs support easy and efficient access to large data sets

+ Formal Relational Query Languages

- Two mathematical query languages form the basis for "real" languages (e.g., SQL), and for implementation:
  - Relational Algebra: Procedural, very useful for representing execution plans
    - E.g., $\pi_{\text{salary}}(\sigma_{\text{firstName = "Jone"}} \text{Employee})$
  - Relational Calculus: Let users describe what they want, rather than how to compute it (Non-Procedural, declarative)
    - E.g., `{t.salary | Employee(t) and t.firstName= "Jone"}`

…we focus on relational algebra: understanding RA is a key to understanding SQL and query processing!

+ Relational Algebra

- Basic operations:
  - Projection (π): deletes unwanted columns from relation
  - Selection (σ): selects a subset of rows from relation
  - Set-difference ($-$): finds tuples in table 1, but not in table 2
  - Union ($\cup$): finds tuples that belong to table 1 or table 2
  - Cross-product ($\times$): allows us to combine two relations
  - Rename (ρ): allows us to rename a relation and/or its attributes
- Additional operations:
  - Intersection ($\cap$), join ($\bowtie$), division (/): Not essential, but (very!) useful
- Each operation returns a relation, and operations can be composed → algebra is closed

…set-based operations

+ Projection $\pi_L(R)$

- Deletes attributes that are not in projection list L
- Schema of result contains exactly the fields in the projection list, with the same names that they had in the (only) input relation
- Projection operator eliminates duplicates!

Maker	Model_No
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
MD	DC10
MD	DC9

$\pi_{Maker}(Plane)$

Maker
Airbus
MD

+ Selection $\sigma_c(R)$

- Selects tuples that satisfy a selection condition c
- Schema of result identical to schema of (only) input relation
- A condition c has the form: Term Op Term
  - where Term is an attribute name or a constant
  - Op is one of <, >, =, ≠, etc.
- (C1 ∧ C2), (C1 ∨ C2), (¬ C1) are conditions where C1 and C2 are conditions
  - ∧ means AND
  - ∨ means OR
  - ¬ means NOT

… row/records/tuples are used without differences

+ Selection Example

Plane

Maker	Model_No
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
MD	DC10
MD	DC9

$\sigma_{\text{Maker="MD"}}(\text{Plane})$

Maker	Model_No
MD	DC10
MD	DC9

- No duplicates in result! (Why?)
- The resulting relation can be the input for another relational algebra operation! (Operator composition)

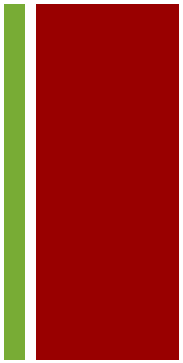
Plane

Maker	Model_No
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
MD	DC10
MD	DC9

$\pi_{\text{Model\_No}}(\sigma_{\text{Maker="MD"}}(\text{Plane}))$

Model_No
DC10
DC9

Set Operations



- Union, Intersection, Set-Difference
- These three operations take two input relations, which must be union-compatible:
  - Same number of fields
  - Corresponding fields have the same type
- Output is a single relation (that does not contain duplicates)

Set Operations - Union

- Plane1 ∪ Plane2

Maker	Model_No
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
MD	DC10
MD	DC9

∪

Maker	Model_No
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

=

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

Set Operations – Set Difference

- Plane1 — Plane2
  - Contains records that appear in Plane1 but not Plane2

Maker	Model_No
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
MD	DC10
MD	DC9

—

Maker	Model_No
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

=

Maker	Model_No
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340

Set Operations - Intersection

- Plane1 ∩ Plane2
  - Contains records that appear in both tables

Maker	Model_No
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
MD	DC10
MD	DC9

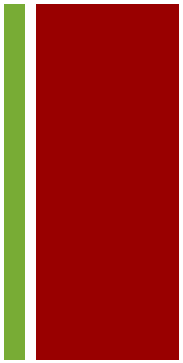
∩

Maker	Model_No
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

=

Maker	Model_No
MD	DC9
MD	DC10

+ Intersection is not Primitive



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

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

+ Cartesian Product

Combines each row of one table with every row of another table

`CanFly` × `Plane`

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

×

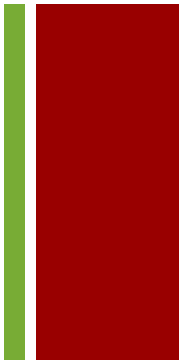
Maker	Model_No
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

=

Emp_No	Model_No	Maker	Model_No
1001	B727	Airbus	A310
1001	B727	Airbus	A320
1001	B727	Airbus	A330
1001	B727	Airbus	A340
1001	B727	Boeing	B727
1001	B727	Boeing	B747
1001	B727	Boeing	B757
1001	B727	MD	DC10
1001	B727	MD	DC9
1001	B747	Airbus	A310
1001	B747	Airbus	A320
1001	B747	Airbus	A330
1001	B747	Airbus	A340
1001	B747	Boeing	B727
1001	B747	Boeing	B747
1001	B747	Boeing	B757
1001	B747	MD	DC10
1001	B747	MD	DC9
1001	B727	Airbus	A310
1001	B727	Airbus	A320
...	...	...	...

9 x 9 = 81 tuples!

+ Join



- Generating all possible combinations of tuples is not usually meaningful
- In the previous example, it makes more sense to combine each tuple of `CanFly` with the corresponding record of the `Plane`
- Join is a cartesian product followed by a selection:

$$R1 \bowtie_c R2 = \sigma_c(R1 \times R2)$$

- Sometimes we use the word JOIN instead of symbol $\bowtie$
- Types of joins:
  - θ-join: arbitrary conditions in the selection
  - Equijoin: all conditions are equalities
  - Natural join: combines two relations on the equality of the attributes with the same names
- Both equijoin and natural join project only one of the redundant attributes

+ Natural Join Example

$$\texttt{CanFly} \bowtie \texttt{Plane}$$

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

$\bowtie$

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

=

Emp_No	Model_No	Maker
1003	A310	Airbus
1002	A320	Airbus
1002	A340	Airbus
1001	B727	Boeing
1001	B747	Boeing
1002	B757	Boeing
1001	DC10	MD
1002	DC9	MD
1003	DC9	MD

+ θ-Join Example

- We have a `Flight` table that records the Number of the flight, Origin, Destination, Departure and Arrival Time.
- We join this table with itself (self-join) using the condition:

```
Flight1.Dest = Flight2.Origin ∧
Flight1.Arr_Time < Flight2.Dept_Time
```

- What should we get?

Num	Origin	Dest	Dep_Time	Arr_Time
334	ORD	MIA	12:00	14:14
335	MIA	ORD	15:00	17:14
336	ORD	MIA	18:00	20:14
337	MIA	ORD	20:30	23:53
394	DFW	MIA	19:00	21:30
395	MIA	DFW	21:00	23:43

$\bowtie_c$

Num	Origin	Dest	Dep_Time	Arr_Time
334	ORD	MIA	12:00	14:14
335	MIA	ORD	15:00	17:14
336	ORD	MIA	18:00	20:14
337	MIA	ORD	20:30	23:53
394	DFW	MIA	19:00	21:30
395	MIA	DFW	21:00	23:43

+ θ-Join Example (cont)

- `Flight1.Dest = Flight2.Origin ∧ Flight1.Arr_Time < Flight2.Dept_Time`

Flight1.Num	Flight1.Origin	Flight1.Dest	Flight1.Dep_Time	Flight1.Arr_Time	Flight2_1.Num	Flight2.Origin	Flight2.Dest	Flight2.Dep_Time	Flight2.Arr_Time
334	ORD	MIA	12:00	14:14	335	MIA	ORD	15:00	17:14
335	MIA	ORD	15:00	17:14	336	ORD	MIA	18:00	20:14
336	ORD	MIA	18:00	20:14	337	MIA	ORD	20:30	23:53
334	ORD	MIA	12:00	14:14	337	MIA	ORD	20:30	23:53
336	ORD	MIA	18:00	20:14	395	MIA	DFW	21:00	23:43
334	ORD	MIA	12:00	14:14	395	MIA	DFW	21:00	23:43

What happens if we add the condition Flight1.Origin ≠ Flight2.Dest

+ Renaming ρ

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

  $\rho(R'(N_1 \rightarrow N'_1, N_n \rightarrow N'_n), R)$

  - The new relation `R'` has the same instance as `R`, but its schema has attribute N'_i instead of attribute N_i

    ```
    ρ(Staff(Name → Family_Name, Salary →
    Gross_salary), Employee)
    ```

  - Necessary for cartesian product or self join

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 Using Basic Operations

- 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

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

+ Example Division

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

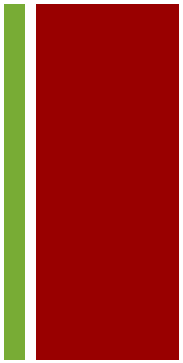
$$\texttt{CanFly} \;/\; \pi_{\texttt{Model\_No}}(\sigma_{\texttt{Maker='MD'}}\ \texttt{Plane})$$

Emp_No	Model_No
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

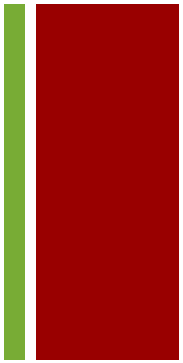
Emp_No
1003

+ Additional Operators - Outer Join



- An extension of the join operation that avoids loss of information
- Computes the join and then adds tuples from one relation that do not match tuples in the other relation to the result of the join
- Uses null values in left- or right- outer join:
  - null signifies that the value is unknown or does not exist
  - All comparisons involving null are false by definition

+ Outer Join - Example



Relation *loan*

branch-name	loan-number	amount
Downtown	L-170	3000
Perryridge	L-260	1700
Redwood	L-230	4000

Relation *borrower*

cust-name	loan-number
Jones	L-170
Smith	L-230
Hayes	L-155

+ Outer Join - Example

loan

branch-name	loan-number	amount
Downtown	L-170	3000
Perryridge	L-260	1700
Redwood	L-230	4000

borrower

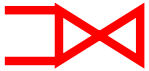
cust-name	loan-number
Jones	L-170
Smith	L-230
Hayes	L-155

Loan ⋈ Borrower

branch-name	loan-number	amount	cust-name
Downtown	L-170	3000	Jones
Redwood	L-230	4000	Smith

Join returns only the matching (or "good") tuples
The fact that loan L-260 has no borrower is not explicit in the result
Hayes has borrowed an non-existent loan L-155 is also undetected

+ Left Outer Join -Example

Left outer join: Loan  borrower

Keep the entire left relation (Loan) and fill in information from the right relation, use null if information is missing.

branch-name	loan-number	amount	cust-name
Downtown	L-170	3000	Jones
Perryridge	L-260	1700	null
Redwood	L-230	4000	Smith

Right and Full Outer Join - Example

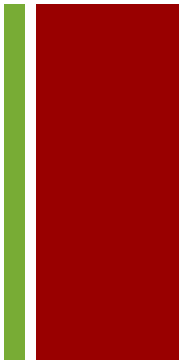
Loan  Borrower

branch-name	amount	cust-name	loan-number
Downtown	3000	Jones	L-170
Redwood	4000	Smith	L-230
null	null	Hayes	L-155

Loan  borrower

branch-name	amount	cust-name	loan-number
Downtown	3000	Jones	L-170
Redwood	4000	Smith	L-230
Perryridge	1700	null	L-260
null	null	Hayes	L-155

+ Summary



- Procedural language for querying databases: Relational Algebra (RA) to obtain data from relational databases
- Five basic operations: π, σ, -, $\cup$, x, p
- Additional operations:
  - Intersection, join, division: Not essential, but useful for querying databases
- The operators take one or two relations as inputs and produce a new relation as a result
- RA is important for studying query optimization