Introduction to Database Systems

Relational Algebra

Lee Mong Li

National University of Singapore

Formal Query Languages

Relational Calculus

- Declarative query language
- Specifies the properties of query answers

Relational Algebra

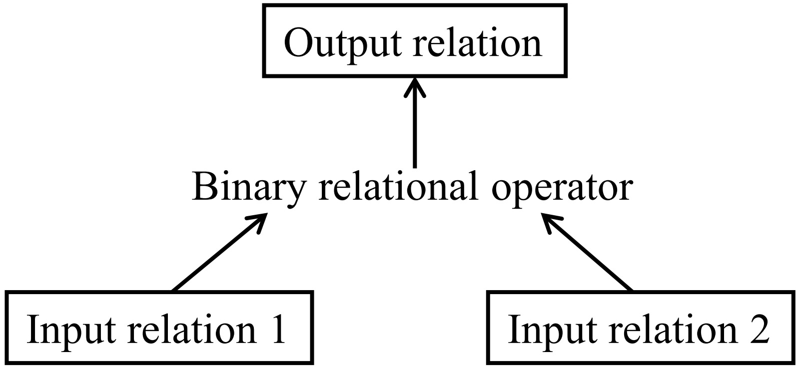
- Procedural or operational query language
- Specifies how to compute query answers
- *SQL queries are executed in terms of relational algebra operations*

Relational Algebra

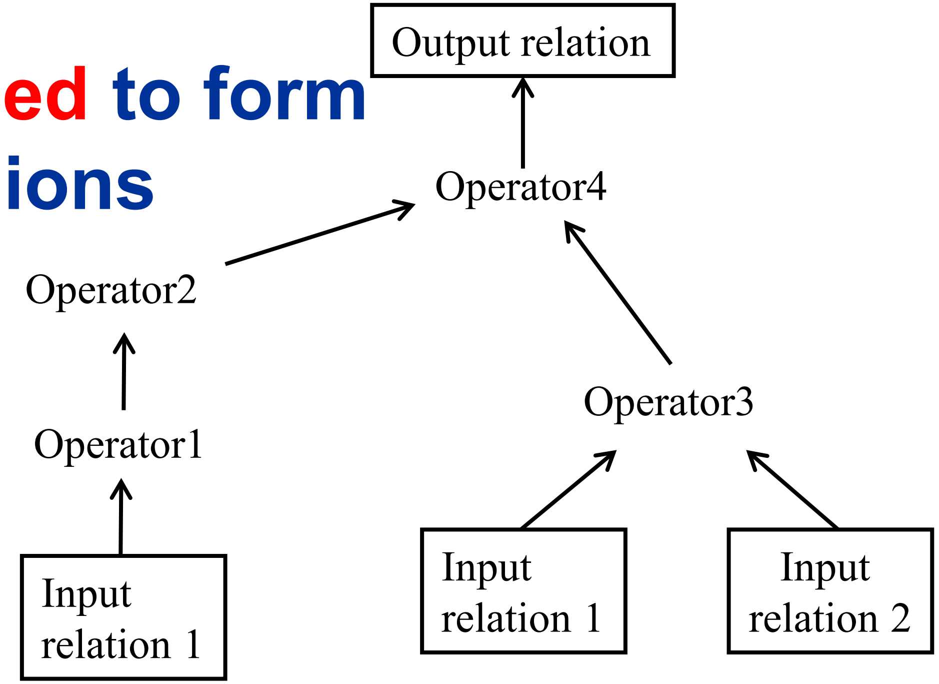
- A formal language for asking queries on relations
- A query is composed of a collection of operators called relational operators
- Each operator takes one or two relations as input and computes an output relation
- Unary operators: selection, projection, etc
- Binary operators: union, intersect, difference, join, etc

Closure Property

- Relations are **closed** under relational operators



- Operators can be **composed** to form relational algebra expressions



Relational Algebra Operators

Binary operators

- Union $\cup$
- Intersection $\cap$
- Difference —
- Cross-product $\times$
- Join $\bowtie_C$
- Division /

Unary operators

- Selection σ
- Projection π
- Renaming ρ

Operators	SQL
σ_C(R)	SELECT * FROM R WHERE *C*
π_L (R)	SELECT DISTINCT *L* FROM R
R $\cup$ S	SELECT * FROM R UNION SELECT * FROM S
R $\cap$ S	SELECT * FROM R INTERSECT SELECT * FROM S
R — S	SELECT * FROM R EXCEPT SELECT * FROM S
R $\times$ S	SELECT * FROM R CROSS JOIN S
R $\bowtie_C$ S	SELECT * FROM R JOIN S ON *C*
R $\bowtie$ S	SELECT * FROM R NATURAL JOIN S

Example Airline Database

Employee

eNumber	name	salary
1006	Clark	150000
1005	Gates	5000000
1001	Jones	50000
1002	Peter	45000
1004	Phillips	25000
1003	Rowe	35000
1007	Warnock	500000

Plane

maker	mNumber
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

CanFly

eNumber	mNumber
1001	B727
1001	B747
1001	DC10
1002	A320
1002	A340
1002	B757
1002	DC9
1003	A310
1003	DC9
1003	DC10

Assigned

eNumber	date	fNumber
1001	Nov 1	100
1001	Oct 31	100
1002	Nov 1	100
1002	Oct 31	100
1003	Oct 31	100
1003	Oct 31	337
1004	Oct 31	337
1005	Oct 31	337
1006	Nov 1	991
1006	Oct 31	337

Projection $\pi_L(R)$

```
SELECT DISTINCT eNumber, fNumber
FROM Assigned
```

- Keeps vertical slices of a relation R according to a list L of attributes (i.e. *a list of columns*) of R
- Duplicate records are removed in output relation

Assigned

eNumber	date	fNumber
1001	Nov 1	100
1001	Oct 31	100
1002	Nov 1	100
1002	Oct 31	100
1003	Oct 31	100
1003	Oct 31	337
1004	Oct 31	337
1005	Oct 31	337
1006	Nov 1	991
1006	Oct 31	337

$\pi_{eNumber,\ fNumber}$ (Assigned)

eNumber	fNumber
1001	100
1002	100
1003	100
1003	337
1004	337
1005	337
1006	991
1006	337

Selection $\sigma_C(R)$

SELECT * FROM Employee
WHERE salary < 100000

- Selects tuples of a relation R satisfying condition c
- c is any Boolean expression involving tuples in R
- Logical operators: $\wedge$, $\vee$, $\neg$
- Comparison operators: =, ≠, >, ≥, ≤, <

Employee

eNumber	name	salary
1006	Clark	150000
1005	Gates	5000000
1001	Jones	50000
1002	Peter	45000
1004	Phillips	25000
1003	Rowe	35000
1007	Warnock	500000

$\sigma_{salary<100000}(Employee)$

eNumber	name	salary
1001	Jones	50000
1002	Peter	45000
1004	Phillips	25000
1003	Rowe	35000

Selection Conditions

- **Selection condition is a boolean combination of terms**
- **A term is one of the following forms:**
  - attribute op constant
  - attribute_1 op attribute_2
  - $\text{term}_1 \wedge \text{term}_2$
  - $\text{term}_1 \vee \text{term}_2$
  - $\neg\ \text{term}_1$
  - (term_1)

op $\in \{ =, \neq, >, \geq, \leq, < \}$

- **Operator precedence: (), op, $\neg$, $\wedge$, $\vee$**

Example

Employee

eNumber	name	salary
1006	Clark	150000
1005	Gates	5000000
1001	Jones	50000
1002	Peter	45000
1004	Phillips	25000
1003	Rowe	35000
1007	Warnock	500000

$\sigma_{salary > 100000 \land \neg (name = \text{'Gates'})}(Employee)$

eNumber	name	salary
1006	Clark	150000
1007	Warnock	500000

```
SELECT * FROM Employee
WHERE salary > 100000
AND name <> 'Gates'
```

Remark: Composability

- **The result of a query is a relation**

$$\sigma_{salary < 50000} \text{ (Employee)}$$

$$\pi_{name,\ salary} (\sigma_{salary < 50000} \text{ (Employee)})$$

Remark: Commutativity

- **Change the order of the operations**

$$\pi_{name,\ salary}(\sigma_{salary < 50000}\ (Employee))$$

$$\sigma_{salary < 50000}(\pi_{name,\ salary}\ (Employee))$$

- **But can we always do this?**

No. Eg, if i'm only interested in the name of employees (cancel the salary in pi sign), then output will be different

Remark: SQL

$$\pi_{name,\ salary}(\sigma_{salary < 50000}\ (Employee))$$

```
SELECT DISTINCT name, salary
FROM employee
WHERE salary < 50000
```

Set Operations

Union: $R_1 \cup R_2 = \{ t \mid t \in R_1 \vee t \in R_2 \}$

Intersection: $R_1 \cap R_2 = \{ t \mid t \in R_1 \wedge t \in R_2 \}$

Set-difference: $R_1 — R_2 = \{ t \mid t \in R_1 \text{ and } \neg(t \in R_2) \}$

- **Input relations R_1 and R_2 must be union compatible**
  - They have the same number of attributes
  - Corresponding attributes have the same domains (but not necessarily use the same attribute name

domain = data type

- **Schema of the result of R op S is identical to the schema of R, where op $\in \{ \cup, \cap, — \}$**

Union (Example)

Plane1

maker	mNumber
Airbus	A310
Airbus	A320
Airbus	A330
Boeing	B747
Boeing	B757

Plane2

maker	mNumber
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
MD	DC10
MD	DC9

$Plane_1 \cup Plane_2$

maker	mNumber
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

```
SELECT *
FROM Plane1
UNION
SELECT *
FROM Plane2
```

What about duplicates?

Union (Example)

- **Find all the customer and restaurant names**

Restaurants (rname: VARCHAR(50), area: VARCHAR(10))
Customers (cname: VARCHAR(50), area: VARCHAR(10))

```
SELECT *
FROM Customers C
UNION
SELECT *
FROM Restaurants R;
```

Intersection (Example)

Plane1

maker	mNumber
Airbus	A310
Airbus	A320
Airbus	A330
Boeing	B747
Boeing	B757

Plane2

maker	mNumber
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
MD	DC10
MD	DC9

$$\text{Plane}_1 \cap \text{Plane}_2$$

maker	mNumber
Airbus	A330
Boeing	B747

```
SELECT *
FROM Plane1
INTERSECT
SELECT *
FROM Plane2
```

What about duplicates?

Intersection (Example)

- Find the emails of students in the computer science department owning a book with ISBN '978-0684801520'.

```
SELECT T1.email
FROM Student T1
WHERE T1.department = 'CS'
INTERSECT
SELECT T2.owner AS email
FROM Copy T2
WHERE T2.ISBN = '978-0684801520';
```

Difference (Example)

Plane1

maker	mNumber
Airbus	A310
Airbus	A320
Airbus	A330
Boeing	B747
Boeing	B757

Plane2

maker	mNumber
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
MD	DC10
MD	DC9

$Plane_1 - Plane_2$

maker	mNumber
Airbus	A310
Airbus	A320
Boeing	B757

```
SELECT *
FROM Plane1
MINUS (EXCEPT)
SELECT *
FROM Plane2
```

What about duplicates?

(Non-Symmetric) Difference

- **Find the mails of students in the computer science department except those owning a book with ISBN '978-0684801520'.**

```
SELECT T1.email
FROM Student T1
WHERE T1.department='CS'
EXCEPT
SELECT T2.owner AS email
FROM Copy T2
WHERE T2.ISBN ='978-0684801520';
```

Cartesian Product

- Combines the tuples of two relations R(A, B, C) and S(X, Y) in all possible ways
- Cross-product: R x S returns a relation with schema (A, B, C, X, Y) defined as

$$R \times S = \{ (a, b, c, x, y) \mid (a, b, c) \in R \wedge (x, y) \in S \}$$

Cartesian Product (Example)

CanFly × Plane

90 tuples!

CanFly

eNumber	mNumber
1001	B727
1001	B747
1001	DC10
1002	A320
1002	A340
1002	B757
1002	DC9
1003	A310
1003	DC9
1003	DC10

Plane

maker	mNumber
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

eNumber	mNumber	maker	mNumber
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
…	…	…	…

```
SELECT *
FROM CanFly, Plane
```

Condition Join (θ-Join)

- Combines the tuples of two relations that verify a condition

- Cross product followed by selection

$$R_1 \bowtie_c R_2 \equiv \sigma_c (R_1 \times R_2)$$

θ-Join (Example)

$$\text{CanFly} \bowtie_{\text{canFly.mNumber=plane.mNumber}} \text{Plane}$$

CanFly

eNumber	mNumber
1001	B727
1001	B747
1001	DC10
1002	A320
1002	A340
1002	B757
1002	DC9
1003	A310
1003	DC9
1003	DC10

Plane

maker	mNumber
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

eNumber	mNumber	maker	mNumber
1001	B727	Boeing	B727
1001	B747	Boeing	B747
1001	DC10	MD	DC10
1002	A320	Airbus	A320
1002	A340	Airbus	A340
1002	B757	Boeing	B757
1002	DC9	MD	DC9
1003	A310	Airbus	A310
1003	DC9	MD	DC9
1003	DC10	MD	DC10

```
SELECT *
FROM CanFly C, Plane P
WHERE C.mNumber = P.mNumber
```

Natural Join

- Combines two relations on a condition composed only of <u>equalities</u> of attributes with the <u>same name</u> in the first and second relation
- Projects only one of the redundant attributes (since they are equal)

$$R_1 \bowtie R_2$$

Natural Join (Example)

CanFly

eNumber	mNumber
1001	B727
1001	B747
1001	DC10
1002	A320
1002	A340
1002	B757
1002	DC9
1003	A310
1003	DC9
1003	DC10

Plane

maker	mNumber
Airbus	A310
Airbus	A320
Airbus	A330
Airbus	A340
Boeing	B727
Boeing	B747
Boeing	B757
MD	DC10
MD	DC9

CanFly ⋈ Plane

eNumber	mNumber	maker
1001	B727	Boeing
1001	B747	Boeing
1001	DC10	MD
1002	A320	Airbus
1002	A340	Airbus
1002	B757	Boeing
1002	DC9	MD
1003	A310	Airbus
1003	DC9	MD
1003	DC10	MD

Renaming

- **Renaming a relation or its attributes:**

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

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

Renaming (Example)

$\rho(\text{Staff}(\text{salary} \rightarrow \text{wages}), \text{Employee})$

Employee

name	salary	eNumber
Clark	150000	1006
Gates	5000000	1005
Jones	50000	1001
Peter	45000	1002
Phillips	25000	1004
Rowe	35000	1003
Warnock	500000	1007

Staff

name	wages	eNumber
Clark	150000	1006
Gates	5000000	1005
Jones	50000	1001
Peter	45000	1002
Phillips	25000	1004
Rowe	35000	1003
Warnock	500000	1007

```
SELECT name, salary AS wages, eNumber
FROM Employee
```

Complex RA Queries

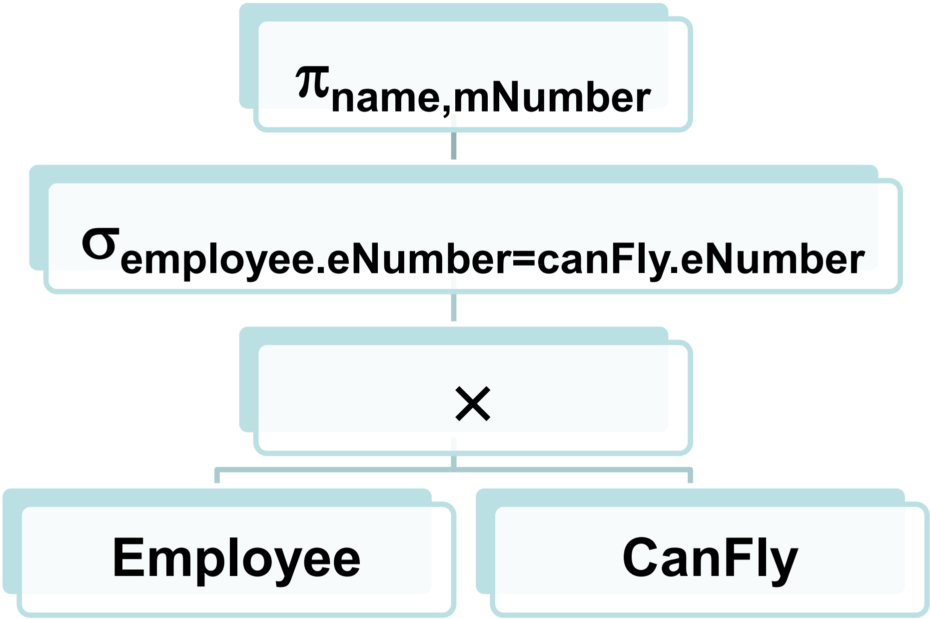
- Find for each employee, his name and the model numbers of the planes he can fly

Employee

eNumber	name	salary
1006	Clark	150000
1005	Gates	5000000
1001	Jones	50000
1002	Peter	45000
1004	Phillips	25000
1003	Rowe	35000
1007	Warnock	500000

CanFly

eNumber	mNumber
1001	B727
1001	B747
1001	DC10
1002	A320
1002	A340
1002	B757
1002	DC9
1003	A310
1003	DC9
1003	DC10



Remark: Project-Select-Join

- Project-Select-Join (PSJ) queries correspond to simple SQL queries:

```
SELECT name, mNumber
FROM Employee, CanFly
WHERE Employee.eNumber = CanFly.eNumber
```

Example

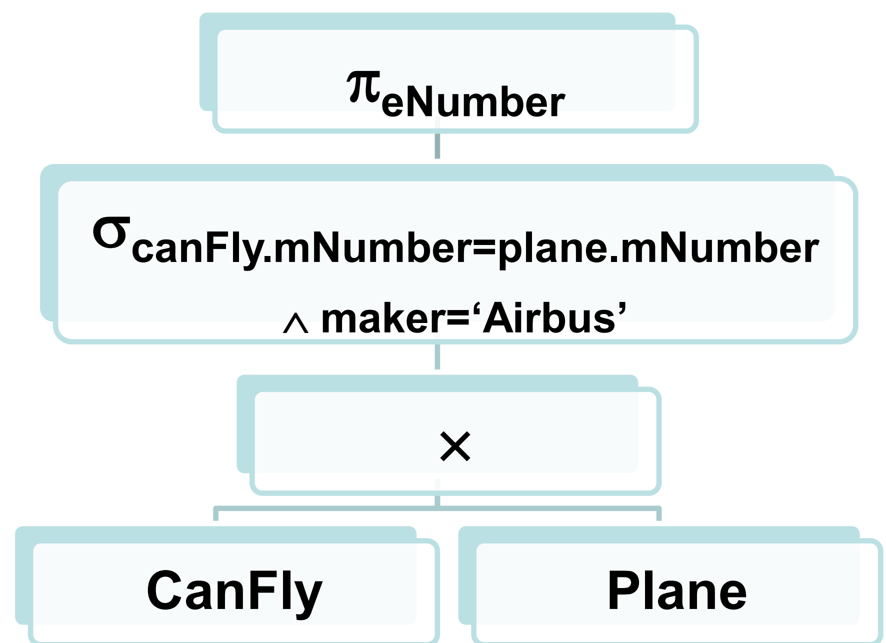
- **Find the employee numbers of employees who can fly Airbus planes**

Employee

eNumber	name	salary
1006	Clark	150000
1005	Gates	5000000
1001	Jones	50000
1002	Peter	45000
1004	Phillips	25000
1003	Rowe	35000
1007	Warnock	500000

CanFly

eNumber	mNumber
1001	B727
1001	B747
1001	DC10
1002	A320
1002	A340
1002	B757
1002	DC9
1003	A310
1003	DC9
1003	DC10



Remark: Rewriting Algebra Expressions

- **Cartesian product with Selection:**

$$\pi_{eNumber}(\sigma_{canFly.mNumber=plane.mNumber \wedge maker='Airbus'}(CanFly \times Plane))$$

- **Selection before Cartesian product:**

$$\pi_{eNumber}(\sigma_{canFly.mNumber=plane.mNumber}(CanFly \times \sigma_{maker='Airbus'}(Plane)))$$

- **θ-join with one condition:**

$$\pi_{eNumber}((CanFly \bowtie_{canFly.mNumber=plane.mNumber} \sigma_{maker='Airbus'}(Plane)))$$

- **θ-join with two conditions:**

$$\pi_{eNumber}(CanFly \bowtie_{canFly.mNumber=plane.mNumber \wedge maker='Airbus'} Plane)$$

Query Equivalence

- **Let Q(d) be the results of query Q on an instance d of a database schema D**
- **Two queries Q1 and Q2 are equivalent, denoted by Q1 ≡ Q2, if for every valid database instance d of D, Q1(d) = Q2(d)**
- **Some equivalence properties**
  - Commutativity of cross-product

    $$R \times S \equiv \pi_{attr(R),\ attr(S)} (S \times R)$$

  - Associativity of cross-product

    $$R \times (S \times T) \equiv (R \times S) \times T$$

  - Commutativity of join

    $$R \bowtie_c S \equiv \pi_{attr(R),\ attr(S)} (S \bowtie_c R)$$

  - Associativity of join

    $$R \bowtie_{c1} (S \bowtie_{c2} T) \equiv (R \bowtie_{c1} S) \bowtie_{c2} T$$

Division

DIVISION NOT TESTED IN EXAM for relational, but still need to know for sql

- **Consider two relations R(x, y) and S(y)**
- **R / S is the set of all x values such that for every y value in S, there is a tuple (x,y) in R**
- **Find all x which are related to every y in S.**
  - Find the employees who can fly all MD planes
  - Find the students who have read all books by “John Piper"
  - Find the actors who have acted in all movies directed by “Steven Spielberg"

Division

CanFly

eNumber	mNumber
1001	B727
1001	B747
1001	DC10
1002	A320
1002	A340
1002	B757
1002	DC9
1003	A310
1003	DC9
1003	DC10

P1

mNumber
DC9

CanFly / P1

eNumber
1002
1003

P2

mNumber
A320
A340

CanFly / P2

eNumber
1002

Division

- Compute *all possible combinations* of column x of R and S: $\pi_x(R) \times S$
- Remove those rows that exist in R: $(\pi_x(R) \times S) - R$
- Keep only the first column of the result. These are the *disqualified* values: $\pi_x((\pi_x(R) \times S) - R)$
- R / S is the column x of R except the disqualified values: $\pi_x(R) - \pi_x((\pi_x(R) \times S) - R)$

Example

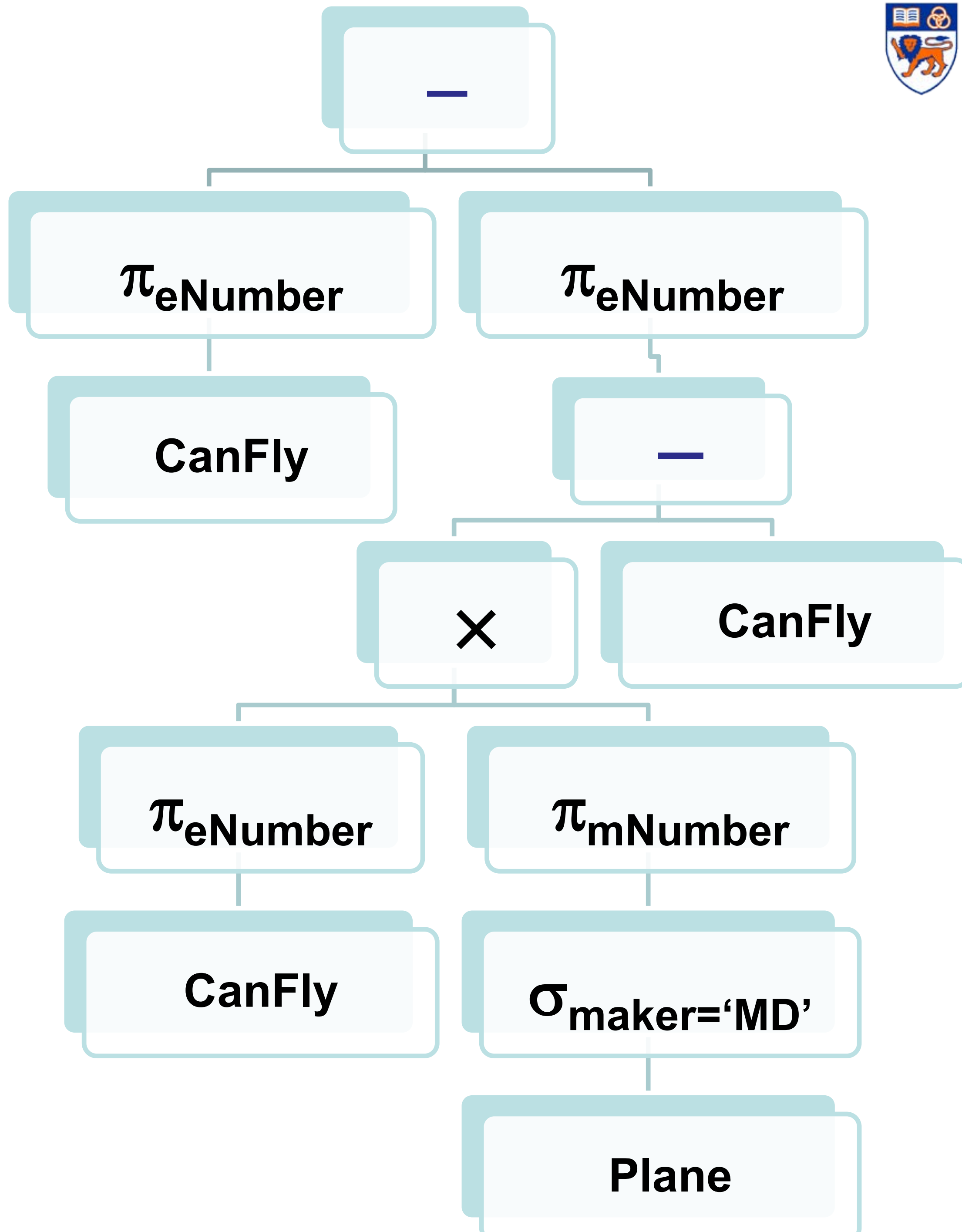
- **Find the employment numbers of employees who can fly all MD planes**

$$\pi_{eNumber}(\text{CanFly}) -$$

$$\pi_{eNumber}((\pi_{eNumber}(\text{CanFly}) \times \pi_{mNumber}(\sigma_{maker='MD'}(\text{Plane}))) - \text{CanFly})$$

- **A complex RA query presented as a single lengthy expression can be unreadable**
  - Difficult to parse deeply nested parenthesis
- **Two ways to improve readability of RA queries**
  - Operator trees
  - Sequence of steps

Operator Tree



Step-by-Step

1. MDPlane = $\pi_{mNumber}(\sigma_{maker='MD'}(Plane))$

mNumber
DC10
DC9

2. R1 = $\pi_{eNumber}(CanFly) \times MDPlane$

eNumber	mNumber
1001	DC9
1001	DC10
1002	DC9
1002	DC10
1003	DC9
1003	DC10

All possible combinations

3. R2 = R1 – CanFly

Remove rows that exist in CanFly

eNumber	mNumber
1001	DC9
1002	DC10

4. R3 = $\pi_{eNumber}(R2)$

Disqualified values

eNumber
1001
1002

5. R4 = $\pi_{eNumber}(CanFly) - R3$

Employees who can fly all MD planes

eNumber
1003

CanFly / MDPlane

Division (SQL)

```
SELECT DISTINCT C1.eNumber
FROM CanFly C1
WHERE NOT EXISTS (
    SELECT *
    FROM Plane P
    WHERE P.maker='MD' AND NOT EXISTS (
            SELECT * FROM CanFly C2
            WHERE C1.eNumber = C2.eNumber
                    AND P.mNumber = C2.mNumber )
)
```

Not exists a MD plane that the employee cannot fly.

CREDITS

The content of this lecture is based on Chapter 4 of the book "Introduction to database Systems" by S. Bressan and B. Catania McGraw Hill publisher