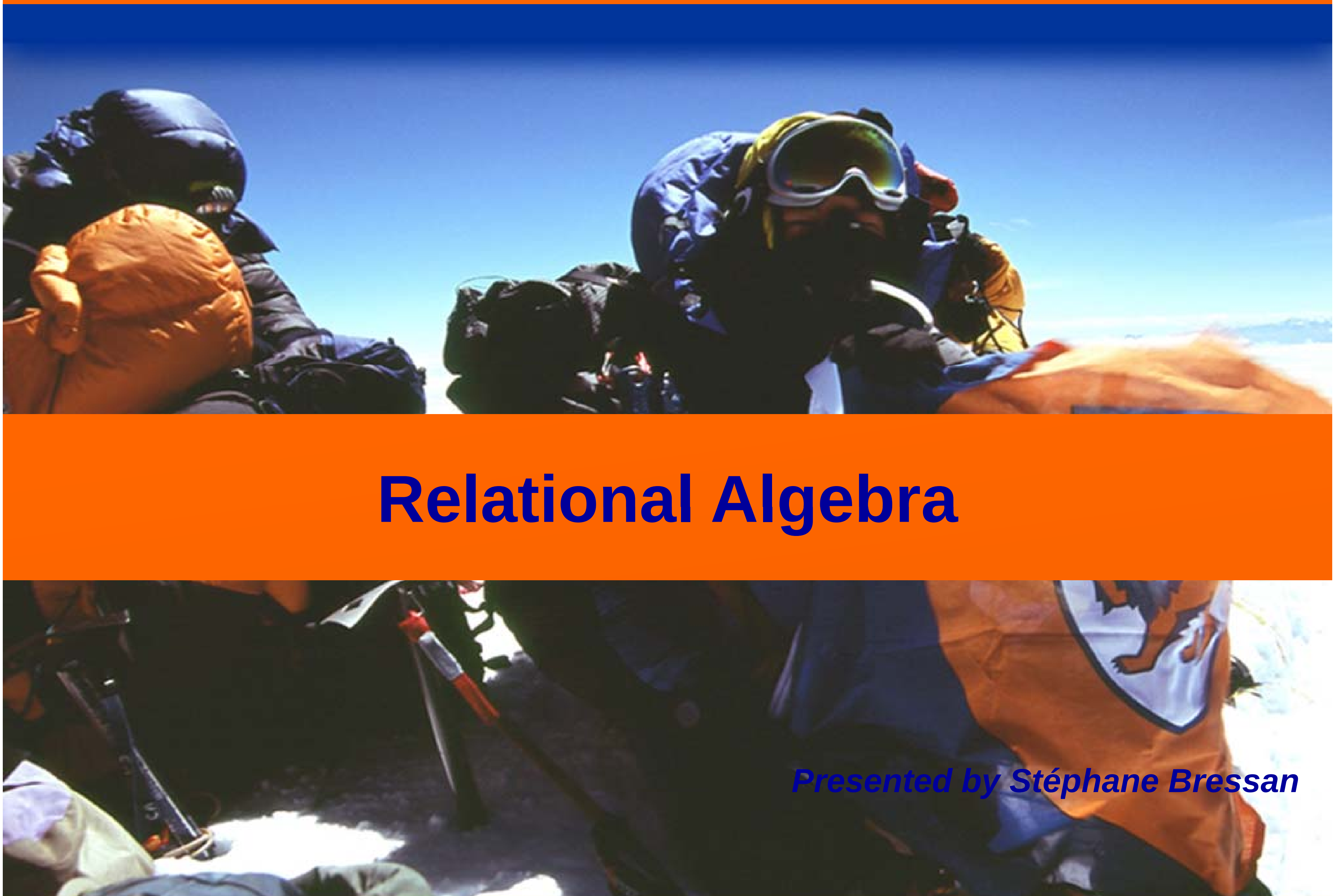


Relational Algebra

Presented by Stéphane Bressan

Relational Query Languages

- Two mathematical Query Languages form the basis for practical languages like SQL:
  - Relational Algebra: Operational, useful for representing execution plans
  - Relational Calculus: Declarative: Describe what you want, rather than how to compute it.
- Query languages are NOT programming languages:
  - Not expected to be Turing complete

Operations (Operators)

- Operations on a single relation
  - selection σ, projection π
- Usual set operations (relations are sets):
  - union $\cup$, intersection $\cap$, and difference $-$ (non-symmetric)
- Operations combining two or more relations
  - Cartesian product $\times$, join $\bowtie$ and natural join $\bowtie_n$
- A renaming operation ρ
- A division operation /

Example: 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

Example: plane

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

Example: canFly

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

Example: 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

Keeps vertical slices of a relation according to a list L of attributes (i.e. *a list of columns*) of the relation R:

$$\pi_L(R) = \{t \mid \exists\, t_1 \\ (t_1 \in R \wedge_{A \in L}\ t.A = t_1.A)\}$$

Projection (Example)

$\pi_{\text{eNumber,, fNumber}}$**(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 (Result)

$\pi_{\text{eNumber,, fNumber}}$(assigned)

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

Projection (SQL)

```
SELECT DISTINCT eNumber, fNumber
FROM assigned
```

Selection

Selects the t-uples of a relation verifying a condition c:

$$\sigma_c(R) = \{t \mid t \in R \wedge c\}$$

c is any Boolean expression ($\wedge$, $\vee$ $\neg$) involving t ($<$, $=$, $>$, $\neq$, $\leq$, $\geq$)

Selection (Example)

$\sigma_{\text{salary<100000}}(\text{employee})$

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

Selection (Result)

$\sigma_{\text{salary}<100000}(\text{employee})$

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

Selection (SQL)

```sql
SELECT *
FROM employee
WHERE salary < 100000
```

Selection (Example)

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

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

Selection (Result)

$\sigma_{salary>100000 \wedge \neg(name='Gates')}(employee)$

name	salary	eNumber
Clark	150000	1006
Warnock	500000	1007

Selection (SQL)

```
SELECT *
FROM employee
WHERE salary > 100000
AND name <> ‘Gates’
```

Remark: Composability

The result of a query is a relation

$\sigma_{salary < 50000}$ (employee)

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

Remark: Commutativity

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

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

Can we always do this?

Remark: SQL

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

SELECT DISTINCT name, salary
FROM employee
WHERE salary < 50000

Projection

Selection

Union, Intersection, Set-difference

- $R_1 \cup R_2 = \{ t \mid t \in R_1 \vee t \in R_2 \}$
- $R_1 \cap R_2 = \{ t \mid t \in R_1 \wedge t \in R_2 \}$
- $R_1 - R_2 = \{ t \mid t \in R_1 \text{ and } \neg(t \in R_2) \}$

The relations R_1 and R_2 must be

union compatible:

- Same number of attributes
- Corresponding attributes have the same type (but not necessarily the same name)

Union (Example)

$plane_1 \cup plane_2$

$plane_1$

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

$plane_2$

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

Union (Result)

$plane_1 \cup plane_2$

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

Union (SQL)

```
SELECT *
FROM plane1
UNION
SELECT *
FROM plane2
```

What about duplicates?

Intersection (Example)

$plane_1 \cap plane_2$

$Plane_1$

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

$Plane_2$

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

Intersection (Result)

$plane_1 \cap plane_2$

maker	mNumber
Airbus	A330
Boeing	B747

Intersection (SQL)

```
SELECT *
FROM plane1
INTERSECT
SELECT *
FROM plane2
```

What about duplicates?

Difference (Example)

$plane_1$ — $plane_2$

$Plane_1$

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

$Plane_2$

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

Difference (Result)

$plane_1 — plane_2$

maker	mNumber
Airbus	A310
Airbus	A320
Boeing	B757

Difference (SQL)

```
SELECT *
FROM plane1
EXCEPT
SELECT *
FROM plane2
```

What about duplicates?

Cartesian Product

Combines the t-uples of two relations in all possible ways

$$
\begin{aligned}
R1 \times R2 = \{t \mid \exists\, t_1\, \exists\, t_2\ &(t_1 \in R_1 \wedge t_2 \in R_2 \\
&\wedge_{A \in R1}\ t.A = t_1.A \\
&\wedge_{A \in R2}\ t.A = t2.A)\}
\end{aligned}
$$

Cartesian Product (Example)

canFly × plane

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

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

Cartesian Product (Result)

canFly × plane

90 t-uples

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
…	…	…	…

Cartesian Product (SQL)

```
SELECT *
FROM canFly, plane
```

Join (θ-Join)

Combines the t-uples of two relations that verify a condition

$$R_1 \bowtie_c R_2 = \{t \mid \exists\, t_1 \exists\, t_2$$

$$(t_1 \in R_1 \wedge t_2 \in R_2 \wedge c$$

$$\wedge_{A \in R1}\ t.A = t_1.A$$

$$\wedge_{A \in R2}\ t.A = t2.A)\}$$

$$= \sigma_c\, (R_1 \times R_2)$$

Join (θ-Join) (Example)

$$\text{canFly} \bowtie_{\text{canFly.mNnumber=plane.mNumber}} \text{plane}$$

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

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

Join (Result)

canFly $\bowtie_{\text{canFly.mNnumber=plane.mNumber}}$ plane

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

Join (SQL)

```
SELECT *
FROM canFly c, plane p
WHERE c.mNumber = p.mNumber
```

Equi-join

- Combines two relations on a condition composed only of equalities of attributes of the first and second relation
- Projects only one of the redundant attributes (since they are equal)

$$R1 \bowtie_{E(A1.1=A2.1 \wedge \ldots \wedge A1.n = A2.n)} R2$$

Equi-join (Example)

canFly $\bowtie_{E(canFly.mNnumber=plane.mNumber)}$ plane

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

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

Equi-join (Result)

canFly $\bowtie_{E(canFly.mNnumber=plane.mNumber)}$ 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

Natural Join

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

$$R_1 \bowtie_N R_2$$

Natural Join (Example)

canFly $\bowtie_N$ plane

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

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

Natural Join (Result)

canFly $\bowtie_N$ 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

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

Renaming (Result)

$\rho(staff(salary \rightarrow wages), employee)$

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

Renaming (SQL)

SELECT name, salary AS wages, eNumber

FROM employee

Example

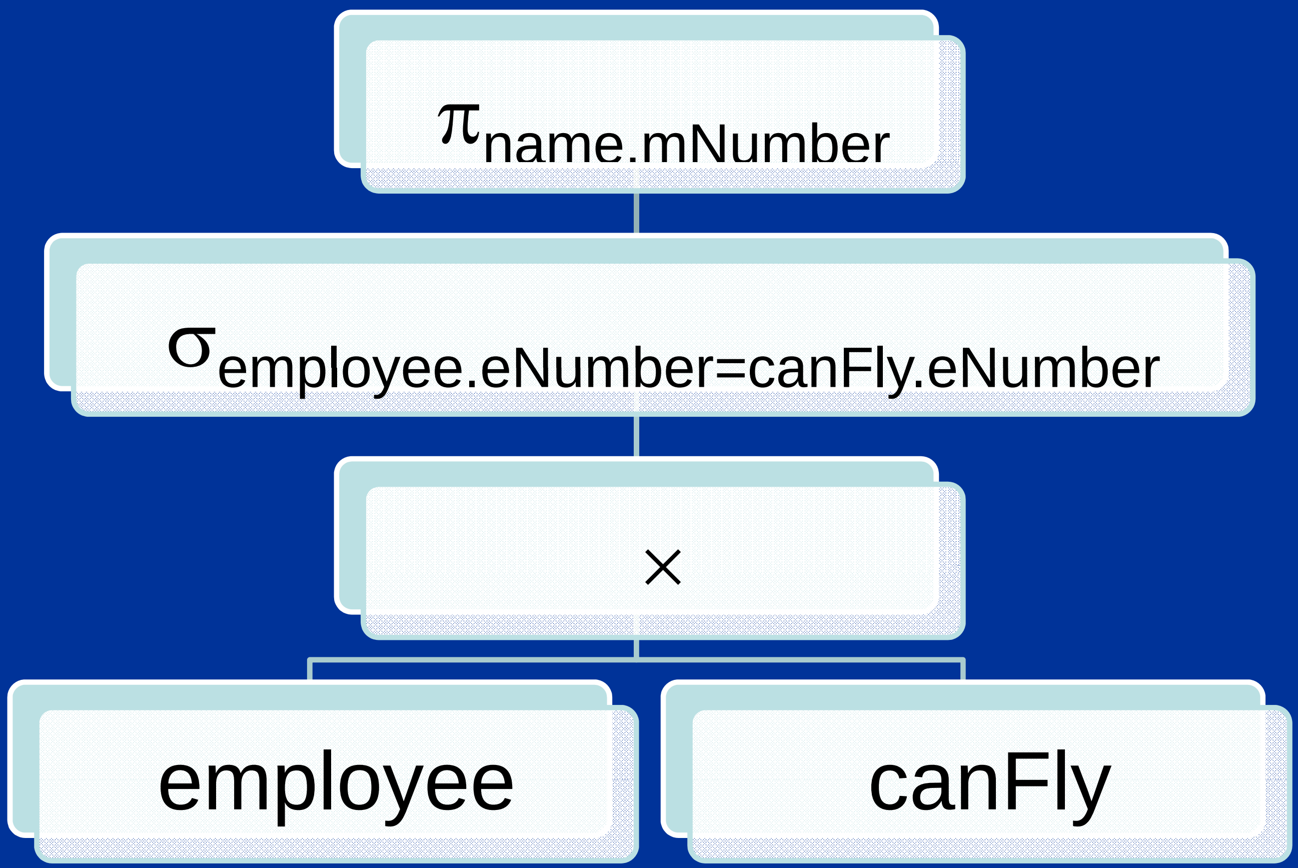
Find for each employee, her name and the model numbers of the planes she can fly

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

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

Example

Find for each employee, her name and the model numbers of the planes she can fly



Remark: Project-Select-Join

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

SELECT name, mNumber ← Projection

FROM employee, canFly ← Cartesian Product

WHERE employee.eNumber=canFly.eNumber

↑ Selection

Remark: Rewriting Algebra Expressions

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

Which one is more efficient?

- $\pi_{\text{eNumber}}(\sigma_{\text{canFly.mNumber=plane.mNumber}}(\text{canFly} \times \sigma_{\text{maker='Airbus'}}(\text{plane})))$
- $\pi_{\text{eNumber}}((\text{canFly} \bowtie_{\text{canFly.mNumber=plane.mNumber}} \sigma_{\text{maker='Airbus'}}(\text{plane})))$
- $\pi_{\text{eNumber}}(\text{canFly} \bowtie_{\text{canFly.mNumber=plane.mNumber} \wedge \text{maker='Airbus'}} \text{plane})$

Example

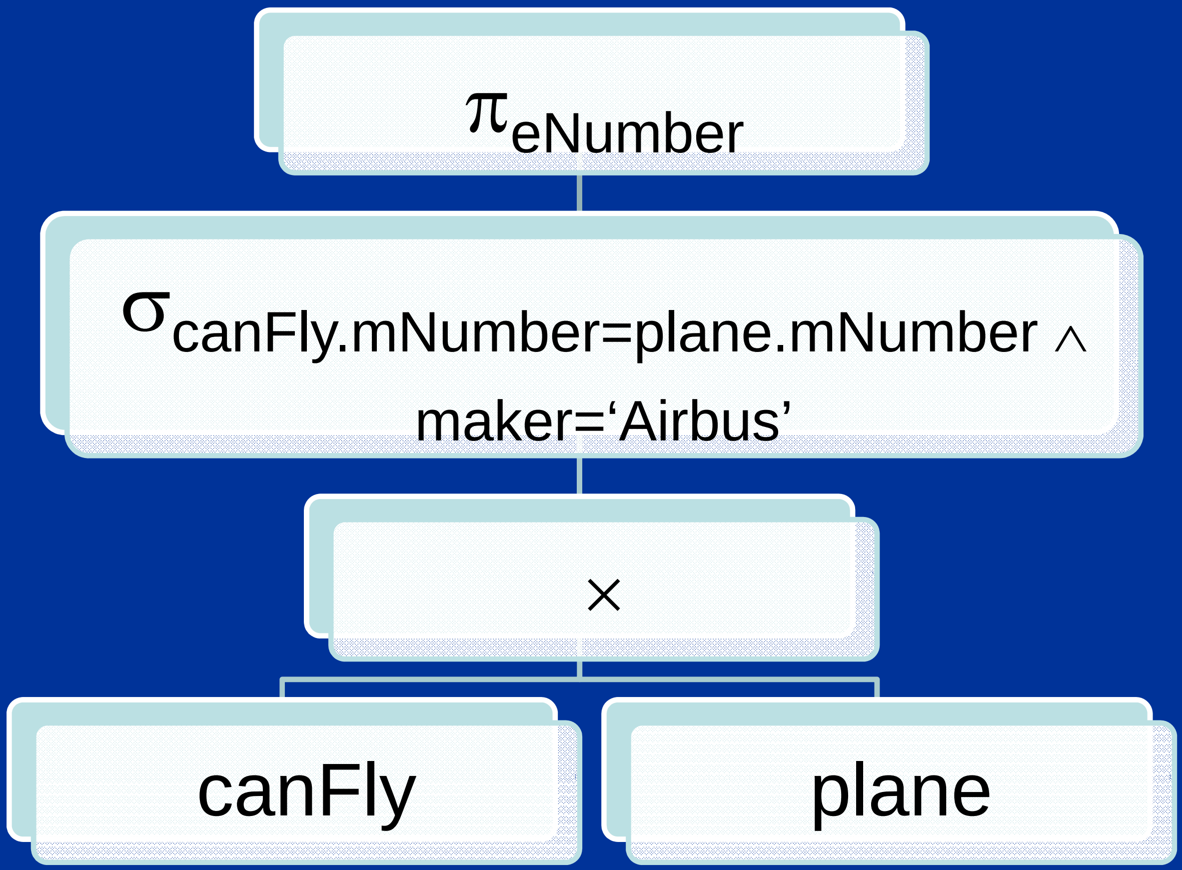
Find the employee numbers of employees who can fly Airbus planes

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

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

Example

Find the employee numbers of employees who can fly Airbus planes



Example

Find the employee numbers of employees who can fly Boeing **or** MD planes

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

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

Example

Find the employee numbers of employees who can fly Boeing **or** MD planes

$\pi_{eNumber}$

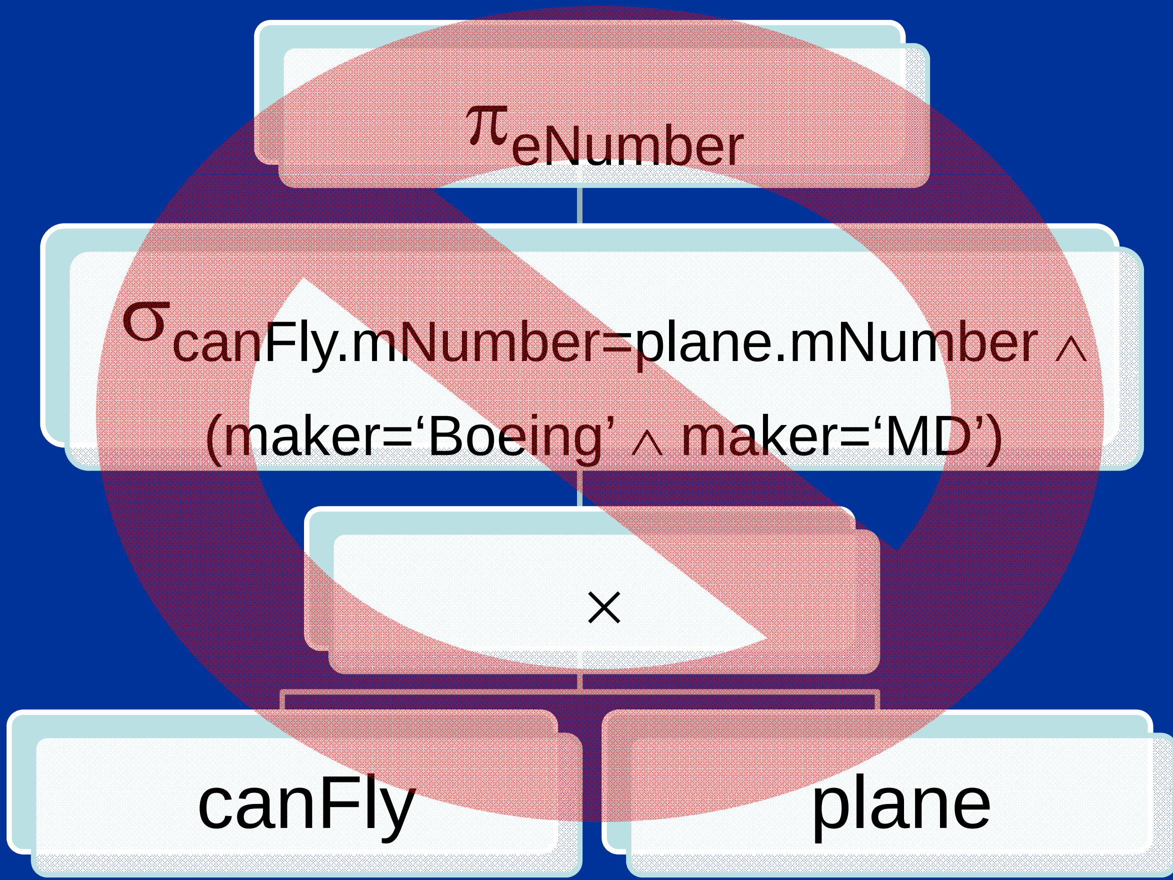
$\sigma_{canFly.mNumber=plane.mNumber \wedge (maker=\text{'Boeing'} \vee maker=\text{'MD'})}$

$\times$

canFly | plane

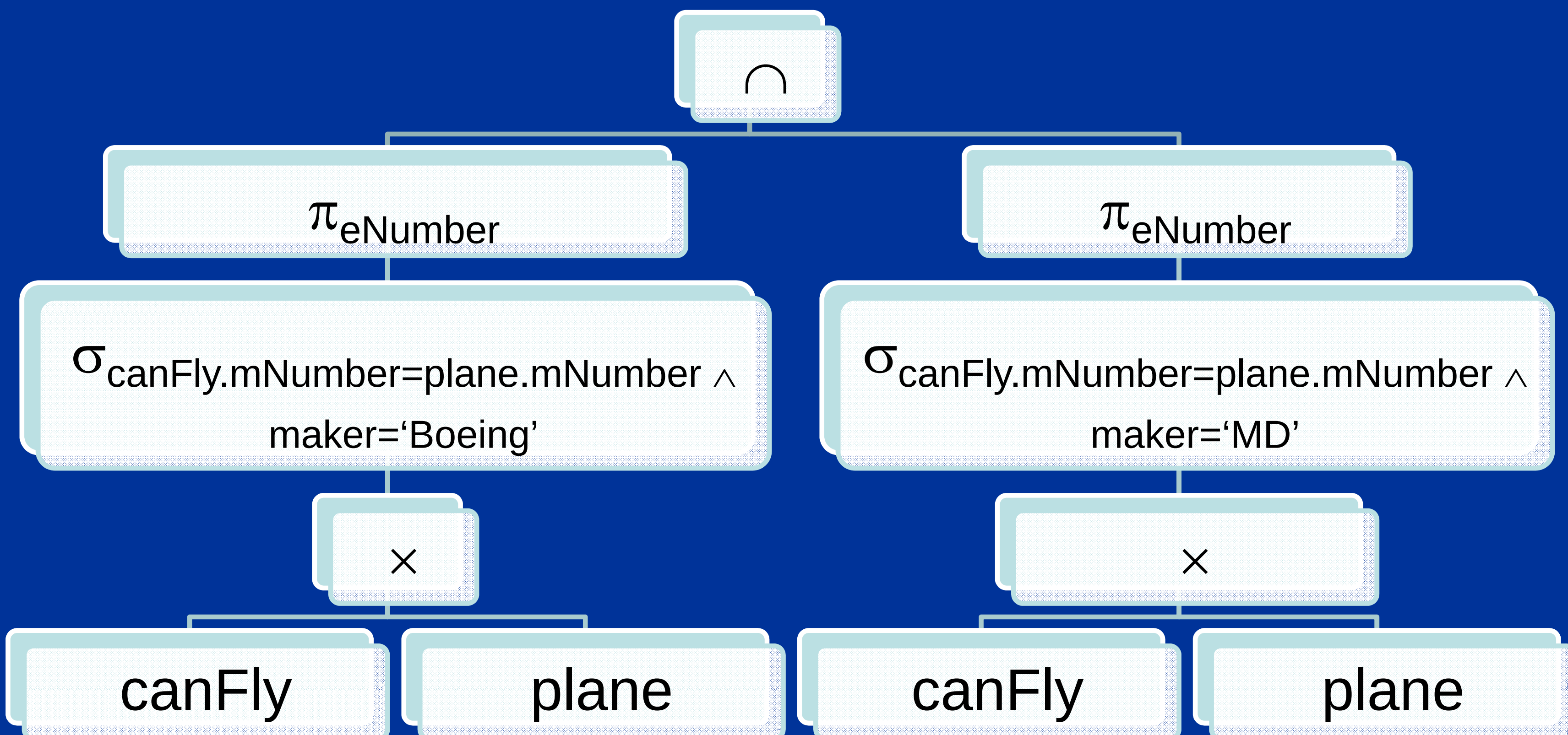
Example

Find the employee numbers of employees who can fly Boeing **and** MD planes



Example

Find the employee numbers of employees who can fly Boeing **and** MD planes



Example

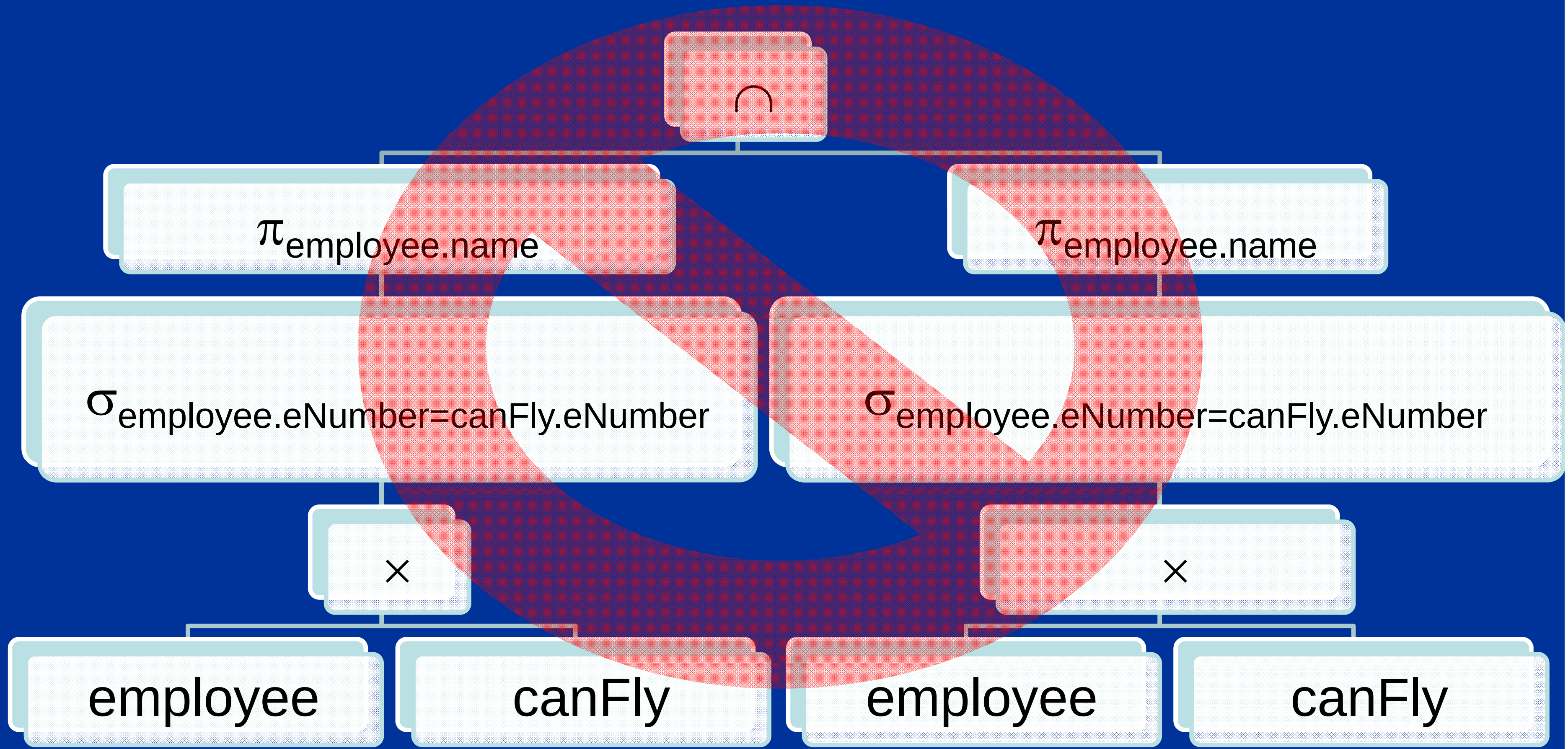
Find the names of employees who can fly two planes or more

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

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

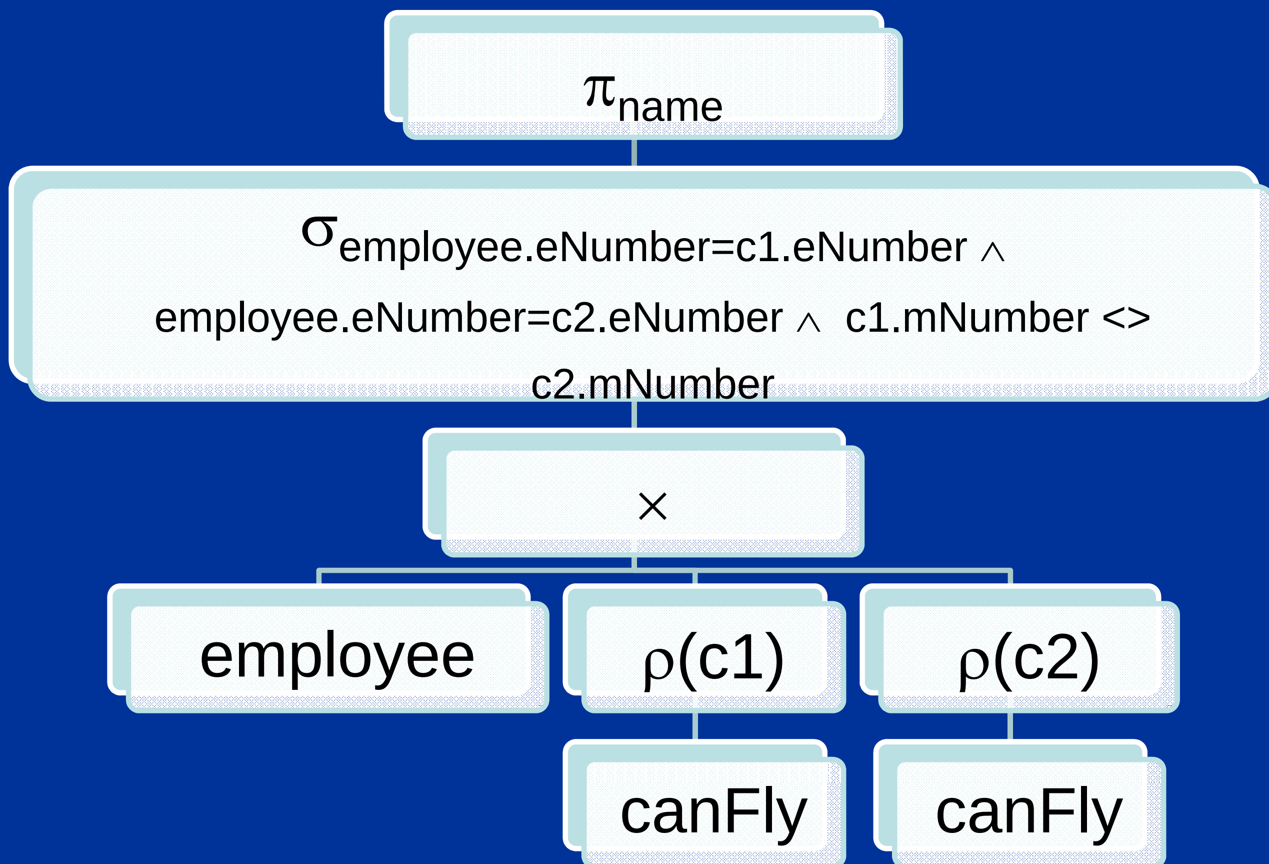
Example

Find the names of employees who can fly two planes or more



Example

Find the names of employees who can fly two planes or more



Example (SQL)

Find the names of employees who can fly two planes or more

```
SELECT employee.name
FROM canFly c1, canFly c2, employee
WHERE c1.eNumber=employee.eNumber
AND c2.eNumber=employee.eNumber
AND c1.mNumber <> c2.mNumber
```

Example (SQL with aggregates)

Find the names of employees who can fly two planes or more (no aggregates in Algebra)

```
SELECT employee.name
FROM canFly, employee
WHERE employee.eNumber=canFly.eNumber
GROUP BY employee.eNumber, employee.name
HAVING COUNT(*) >= 2
```

Division

- Compute all possible combinations of the first column of R_1 and R_1.

$$(\pi_A(R_1) \times R_2)$$

- Then remove those rows that exist in R1

$$(\pi_A(R_1) \times R_2) - R_1$$

- Keep only the first column of the result. These are the ***disqualified*** values

$$\pi_A((\pi_A(R_1) \times R_2) - R_1)$$

- A/B is the first column of R_1 **except** the disqualified values

$$R_1 / R_2 = \pi_A(R_1) - \pi_A((\pi_A(R_1) \times R_2) - R_1)$$

Example

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

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

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

Division (Example)

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

$$\text{canFly} \,/\, \pi_{\text{mNumber}}(\sigma_{\text{maker='MD'}}(\text{plane}))$$

$$\pi_{\text{eNumber}}(\text{canFly}) - \pi_{\text{eNumber}}((\pi_{\text{eNumber}}(\text{canFly}) \times \pi_{\text{mNumber}}(\sigma_{\text{Maker='MD'}}(\text{plane}))) - \text{canFly})$$

Example: step-by-step

$$\pi_{eNumber}(canFly) \times \pi_{mNumber}(\sigma_{Maker=\text{'MD'}}(plane))$$

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

Example: step-by-step

$(\pi_{eNumber}(canFly) \times$
$\pi_{mNumber}(\sigma_{Maker=\text{'MD'}}(plane))) - canFly$

eNumber	mNumber
1001	DC9
1002	DC10

Example: step-by-step

$$\pi_{eNumber}((\pi_{eNumber}(canFly) \times \pi_{mNumber}(\sigma_{Maker='MD'}(plane))) - canFly)$$

eNumber
1001
1002

Example: step-by-step

$\pi_{eNumber}$(canFly) –

$\pi_{eNumber}$(($\pi_{eNumber}$(canFly)

$\times$ $\pi_{mNumber}$($\sigma_{Maker='MD'}$(plane)))–

canFly)

eNumber
1003

Division (SQL)

```
SELECT  DISTINCT c1.eNumber
FROM canFly c1
WHERE NOT EXISTS
        (SELECT c.eNumber, p.mNumber
        FROM canFly c, plane p
        WHERE  p.maker='MD'
                AND c1.eNumber=c.eNumber
        EXCEPT
        SELECT enumber, mNumber
        FROM  canFly)
```

Division (SQL)

```
SELECT DISTINCT c1.eNumber
FROMcanFly 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))
```

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

Animated characters are animated using VocaliseTTS under license from Digital Curiosty

Clipart and media are licensed from Microsoft Office Online Clipart and Media