Data Management and Analysis

Giuseppe Perelli
perelli@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

The content of the slides is authored by Prof. Maria de Marsico, Sapienza University of Rome
Part of the material is taken from the book: Atzeni, Ceri, Paraboschi, Torlone Basi di dati - Modelli e linguaggi di interrogazione , McGraw-Hill, 2002

Universal quantification

- until now, our queries implied the existential quantification:

  $\exists$ (there exists, for some)

- we can find the tuples satisfying a given condition by going through all the tuples and checking to condition on each of them; whenever we find one, we insert it into the result

Universal quantification

Query: customers who made (implied: at least) an order of 100 pieces or more:

order

C#	A#	N-pieces
C1	A1	200
C2	A2	100
C1	A2	50
C4	A3	150
C2	A2	200
C1	A3	100

$\sigma_{\text{N-pieces}>100}$(order)

C#	A#	N-pieces
C1	A1	200
C4	A3	150
C2	A2	200

Universal quantification

- now we want to write queries for the universal quantification:

  ∀ (for all)

  which is equivalent to:

  !∃ (there is none)

Example 1 reloaded

query : names and towns of all the customers who made at least an order of 100 pieces

customer

Name	C#	Town
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

order

C#	A#	N-pieces
C1	A1	100
C2	A2	200
C3	A2	150
C4	A3	200
C1	A2	200
C1	A3	100

$$\pi_{\text{Name, Town}}(\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order}))$$

Example 1 reloaded

Name	C#	Town	A#	N-pieces
Rossi	C1	Roma	A1	100
Rossi	C1	Roma	A2	200
Rossi	C1	Roma	A3	100
Rossi	C2	Milano	A2	200
Bianchi	C3	Roma	A2	150
Verdi	C4	Roma	A3	200

customer $\bowtie$ order

after computing the internal join, the query execution continues by checking all the tuples, one by one, and every time there is a tuple satisfying the condition, it is inserted in the result

Example 1 reloaded

Name	C#	Town	A#	N-pieces
Rossi	C1	Roma	A2	200
Rossi	C2	Milano	A2	200
Bianchi	C3	Roma	A2	150
Verdi	C4	Roma	A3	200

$$\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order})$$

and finally we project:

$$\pi_{\text{Name, Town}}(\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order}))$$

so we have the answer to the query: find the customers who there exist at least an order with more than 100 pieces

Example 1 modified

Query : names and towns of the customers who **ALWAYS** ordered more than 100 pieces

customer

Name	C#	Town
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

order

C#	A#	N-pieces
C1	A1	100
C2	A2	200
C3	A2	150
C4	A3	200
C1	A2	200
C1	A3	100

???$(\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order}))$

Example 1 modified

Name	C#	Town	A#	N-pieces
Rossi	C1	Roma	A1	100
Rossi	C1	Roma	A2	200
Rossi	C1	Roma	A3	100
Rossi	C2	Milano	A2	200
Bianchi	C3	Roma	A2	150
Verdi	C4	Roma	A3	200

customer $\bowtie$ order

the second tuple satisfies the condition:

$\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order})$

… but we cannot insert it into the result, as we should "remember" that Rossi did not order more than 100 pieces for another order

Example 1 modified

Name	C#	Town	A#	N-pieces
Rossi	C1	Roma	A2	200
Rossi	C1	Roma	A1	100
Rossi	C1	Roma	A3	100
Rossi	C2	Milano	A2	200
Bianchi	C3	Roma	A2	150
Verdi	C4	Roma	A3	200

Idea: switch the tuples(?), so the first one satisfies the condition

$$\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order})$$

… but, still, we cannot insert it …. as we will consider other orders of the same customer that do not satisfy the condition

Important: first-order logic

the negation of «**for all»:**

«∀ item the condition is true »

is not «for none»:

«∀ item the condition is false» **<-- no!**

but it is:

«∃ an item for which the condition is false»

the negation of:

«ALL my friends are named Paolo»

is not:

«NONE of my friends is named Paolo»

but it is:

«NONE of my friends is NOT named Paolo»

Solution

- we exploit the double negation
- we invert the condition in the original query
- we find the items that satisfies the inverted condition so, these objects cannot satisfy the "for all" condition
- we remove these items from the solution

Example 1 modified

Name	C#	Town	A#	N-pieces
Rossi	C1	Roma	A1	100
Rossi	C1	Roma	A2	200
Rossi	C1	Roma	A3	100
Rossi	C2	Milano	A2	200
Bianchi	C3	Roma	A2	150
Verdi	C4	Roma	A3	200

customer $\bowtie$ order

$\sigma_{N\text{-}pieces<=100}(\text{customer} \bowtie \text{order})$

Example 1 modified

Name	C#	Town	A#	N-pieces
Rossi	C1	Roma	A1	100
Rossi	C1	Roma	A3	100

$$\sigma_{\text{N-pieces}<=100}(\text{customer} \bowtie \text{order})$$

we project:

$$\pi_{\text{Name, Town}}(\sigma_{\text{N-pieces}<=100}(\text{customer} \bowtie \text{order}))$$

so we obtain the objects that DO NOT satisfy the original query

$$\boxed{\pi_{\text{Name, Town}}(\text{customer} \bowtie \text{order}) - \pi_{\text{Name, Town}}(\sigma_{\text{N-pieces}<=100}(\text{customer} \bowtie \text{order}))}$$

finally, from all the objects we subtract those that do not satisfy the condition AT LEAST one time

Example 1 - note

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

we could think of subtracting from all the customers, but that is not correct, as there can be a customer who did not make any order:

$\pi_{\text{Name, Town}}(\text{customer}) - \pi_{\text{Name, Town}}(\sigma_{\text{N-pieces<=100}}(\text{customer} \bowtie \text{order}))$

so, the correct solution is to subtract from the result of the **unconditioned** relation (all the customers who made at least an order) the result of the conditioned one:

$\pi_{\text{Name, Town}}(\text{customer} \bowtie \text{order}) - \pi_{\text{Name, Town}}(\sigma_{\text{N-pieces<=100}}(\text{customer} \bowtie \text{order}))$

Example 1 modified 2

Query : names and towns of the customers who **never** ordered more than 100 pieces of an article

customer

Name	C#	Town
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

order

C#	A#	N-pieces
C1	A1	100
C2	A2	200
C3	A2	150
C4	A3	200
C1	A2	200
C1	A3	100

$???(\sigma_{\text{N-pieces}<=100}(\text{customer} \bowtie \text{order}))$

Example 1 modified 2

Name	C#	Town	A#	N-pieces
Rossi	C1	Roma	A1	100
Rossi	C1	Roma	A2	200
Rossi	C1	Roma	A3	100
Rossi	C2	Milano	A2	200
Bianchi	C3	Roma	A2	150
Verdi	C4	Roma	A3	200

the first tuple satisfies the condition:

$$\sigma_{\text{N-pieces}<=100}(\text{customer} \bowtie \text{order})$$

… but we cannot insert it …. as Rossi appears also in the next one, which do not satisfy the condition

Example 1 modified 2

Name	C#	Town	A#	N-pieces
Rossi	C1	Roma	A1	100
Rossi	C1	Roma	A2	200
Rossi	C1	Roma	A3	100
Rossi	C2	Milano	A2	200
Bianchi	C3	Roma	A2	150
Verdi	C4	Roma	A3	200

customer $\bowtie$ order

so, we apply the same reasoning as before by first selecting the tuples that we do not want:

$$\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order})$$

Example 1 modified 2

Name	C#	Town	A#	N-pieces
Rossi	C1	Roma	A2	200
Rossi	C2	Milano	A2	200
Bianchi	C3	Roma	A2	150
Verdi	C4	Roma	A3	200

$$\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order})$$

then, we project:

$$\pi_{\text{Name, Town}}(\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order}))$$

finally, we subtract:

$$\pi_{\text{Name, Town}}(\text{customer} \bowtie \text{order}) - \pi_{\text{Name, Town}}(\sigma_{\text{N-pieces}>100}(\text{customer} \bowtie \text{order}))$$

Join with the same table

- there are cases in which we need to compute the join of a table with itself, so we "create" pairs of tuples of the same table

Join with the same table

query: names and codes of the employees who have a salary higher than their supervisors

Employees

Name	C#	Section	Salary	Supervisor#
Rossi	C1	B	100	C3
Pirlo	C2	A	200	C3
Bianchi	C3	A	500	NULL
Verdi	C4	B	200	C2
Neri	C5	B	150	C1
Tosi	C6	B	100	C1

Join with the same table

- the information about the salary of an employee and their supervisor are in different tuples of the same table

- **in order to compare values, they need to be in different columns of the same tuple**

- how should we proceed?

Join with the same table

- we can make a copy of the table and compute the product so we combine the information of all the employees and supervisors together
- now we have a new relation in which the tuples are pairs (employee, supervisor), so we can check conditions
- to do that, we compute the join between Employee and its copy, taking all the tuples in which C# is equal to Supervisor#
- in doing that, we rename the columns, to easily distinguish between the data of employees and supervisors

Example 1

EmployeesC = $\rho_{\text{Name, C\#, Section, Salary, Supervisor\#} \rightarrow \text{CName, CC\#, CSection, Cstip, CSupervisor\#}}$ (Employees)

Name	C#	Section	Salary	Supervisor#	CName	CC#	CSection	CStip	CSupervisor#
Rossi	C1	B	100	C3	Bianchi	C3	A	500	NULL
Pirlo	C2	A	200	C3	Bianchi	C3	A	500	NULL
Verdi	C4	B	200	C2	Pirlo	C2	A	200	C3
Neri	C5	B	150	C1	Rossi	C1	B	100	C3
Tosi	C6	B	100	C1	Rossi	C1	B	100	C3

$\sigma_{\text{Supervisor\#=CC\#}}(\textit{Employees} \times \textit{EmployeesC})$

BE CAREFUL! Using natural join here would not work, as we would combine each employee with themselves (also, there is C3 who does not have a supervisor, so that employee would not be included in the result)

Join with the same table

relational algebra is a formal language, so when we compute the join between two instances of the same relation we could use different strategies, based on different conventions

Join with the same table

EmployeesC = Employees

$\sigma_{\text{Employees.Supervisor\#=EmployeesC.C\#}}(\mathit{Employees} \times \mathit{EmployeesC})$

or:

$\mathbf{\mathit{Employees}} \times \rho_{\mathit{Name, C\#, Section, Salary, Supervisor\#} \rightarrow \mathit{CName, CC\#, CSection, Cstip, CSupervisor\#}}(\mathbf{\mathit{Employees}})$

$\sigma_{\text{Supervisor\#=CC\#}}(\mathit{Employees} \times \rho_{\mathit{Name, C\#, Section, Salary, Supervisor\#} \rightarrow \mathit{CName, CC\#, CSection, Cstip, CSupervisor\#}}(\mathit{Employees}))$

or:

$\mathbf{\mathit{EmployeesC}} = \rho_{\mathit{Name, C\#, Section, Salary, Supervisor\#} \rightarrow \mathit{CName, CC\#, CSection, Cstip, CSupervisor\#}}(\mathbf{\mathit{Employees}})$

$\sigma_{\mathit{Supervisor\#=CC\#}}(\mathit{Employees} \times \mathit{EmployeesC})$

Example 1

- now we just need to apply the condition on the salary and then project:

$$r=(\sigma_{Salary>=CStip}(\sigma_{Supervisor\#=CC\#}(Employees \times EmployeesC)))$$

Name	C#	Section	Salary	Supervisor#	CName	CC#	CSection	CStip	CSupervisor#
Verdi	C4	B	200	C2	Pirlo	C2	A	200	C3
Neri	C5	B	150	C1	Rossi	C1	B	100	C3
Tosi	C6	B	100	C1	Rossi	C1	B	100	C3

Name	C#
Verdi	C4
Neri	C5
Tosi	C6

$$\pi_{Name,\ C\#}(r)$$

Example 2

- **query:** names and codes of the supervisors who gain more money than **all** their employees

again, this is a "for all" condition!

Example 2

- remember example 1, all the employees who have a salary higher than their supervisors: the supervisors who appear in the result are those who should **not** appear in the result!

$r = (\sigma_{Salary>=CStip}(\sigma_{Supervisor\#=CC\#}(Employees \times EmployeesC)))$

Name	C#	Section	Salary	Supervisor#	CName	CC#	CSection	CStip	CSupervisor#
Verdi	C4	B	200	C2	Pirlo	C2	A	200	C3
Neri	C5	B	150	C1	Rossi	C1	B	100	C3
Tosi	C6	B	100	C1	Rossi	C1	B	100	C3

$\pi_{CName,CC\#}(Employees \times EmployeesC) - \pi_{CName,CC\#}(r)$

Bianchi	C3

Example 2

- the same exercise can have multiple correct solution, but be careful with the wrong ones:

$$\pi_{Name,C\#}(Employees) - \pi_{CName,CC\#}(r)$$

it is wrong, as there are employees who are not supervising anyone and those would be excluded from the result

$$(\pi_{Name,Supervisor\#}(Employees)) - \pi_{CName,CC\#}(r)$$

it is wrong, as we are subtracting names and codes of supervisors from names and codes of employees...

correct alternative:

$$\pi_{Name,C\#}((\pi_{Supervisor\#}(Employees) - \pi_{CC\#}(r)) \bowtie Employees)$$

Example 3

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

query: names and codes of the employee(s) who has (have) the highest salary

Employees

Name	C#	Section	Salary
Rossi	C1	B	100
Pirlo	C2	A	200
Bianchi	C3	A	500
Verdi	C4	B	200
Neri	C5	B	150
Tosi	C6	B	100

Example 3

idea: invert the condition, find all the employees whose salary is lower of at least another employee

Employee2 = Employee

$\text{Employee} \bowtie_{\text{Employee.Salary} < \text{Employee2.Salary}} \text{Employee2}$

Employee.Name	Employee.C	Employee.Section	Employee.Salary	Employee2.Name	Employee2.C	Employee2.Section	Employee2.Salary
Rossi	C1	B	100	Pirlo	C2	A	200
Rossi	C1	B	100	Bianchi	C3	A	500
Rossi	C1	B	100	Verdi	C4	B	200
Rossi	C1	B	100	Neri	C5	B	150
Pirlo	C2	A	200	Bianchi	C3	A	500
Verdi	C4	B	200	Bianchi	C3	A	500
Neri	C5	B	150	Pirlo	C2	A	200
Neri	C5	B	150	Bianchi	C3	A	500
Neri	C5	B	150	Verdi	C4	B	200
Tosi	C6	B	100	Pirlo	C2	A	200
Tosi	C6	B	100	Bianchi	C3	A	500
Tosi	C6	B	100	Verdi	C4	B	200
Tosi	C6	B	100	Neri	C5	B	150

Example 3

idea: now get all the codes of those employees, and subtract them from all the employees

$\pi_{Employee.C}$ (Employee $\bowtie_{Employee.Salary < Employee2.Salary}$ Employee2)

π Employee.C (Employee)

Employee.C
C1
C2
C3
C4
C5
C6

-

Employee.C
C1
C2
C4
C5
C6

= C3