# 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

# Multiple relations

- we will see that sometimes it is necessary to split the same information in multiple relations (e.g., to normalize a database)
- normally, the information we would like to retrieve from the database is contained in multiple relations
- so, we need to combine the content of multiple tuples from multiple relations into new tuples

# Cartesian product

- it creates a relation with tuples obtained by combining all the tuples in the first relation with all the tuples in the second relation
- we use the symbol x

$$r_1 x r_2$$

- we use it when the information we need is contained in multiple relations
- but...

# Cartesian product

**Customer**

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

**Order**

| O# | C# | A# | N-pieces |
|---|---|---|---|
| O1 | C1 | A1 | 100 |
| O2 | C2 | A2 | 200 |
| O3 | C3 | A2 | 150 |
| O4 | C4 | A3 | 200 |
| O1 | C1 | A2 | 200 |
| O1 | C1 | A3 | 100 |

query: all the customers and their orders (Customer × *Order*)

# Cartesian product

**Customer**

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

**Order**

| O# | C# | A# | N-pieces |
|---|---|---|---|
| O1 | C1 | A1 | 100 |
| O2 | C2 | A2 | 200 |
| O3 | C3 | A2 | 150 |
| O4 | C4 | A3 | 200 |
| O1 | C1 | A2 | 200 |
| O1 | C1 | A3 | 100 |

first of all, we need to distinguish C# in the first relation from C# in the second relation, so we use the renaming operator $\rho$ to create a copy of Order in which C# is renamed to CC#:

$$OrderR = \rho_{CC\#\,C\#}(Order)$$

# Result

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

| Name | C# | Town | O# | CC# | A# | N-pieces |
|---|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | C1 | A1 | 100 |
| Rossi | C1 | Roma | O2 | C2 | A2 | 200 |
| Rossi | C1 | Roma | O3 | C3 | A2 | 150 |
| Rossi | C1 | Roma | O4 | C4 | A3 | 200 |
| Rossi | C1 | Roma | O1 | C1 | A2 | 200 |
| Rossi | C2 | Milano | O1 | C1 | A1 | 100 |
| … | … | … | | … | … | … |
| Bianchi | C3 | Roma | O3 | C1 | A1 | 100 |
| … | … | … | | … | … | … |
| Verdi | C4 | Roma | O4 | | A3 | 200 |
| … | … | … | | … | … | … |

# Result

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

| Name | C# | Town | O# | CC# | A# | N-pieces |
|---|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | C1 | A1 | 100 |
| Rossi | C1 | Roma | O2 | C2 | A2 | 200 |
| Rossi | C1 | Roma | | C3 | A2 | 150 |
| Rossi | C1 | Roma | | C4 | A3 | 200 |
| Rossi | C1 | Roma | | C1 | A2 | 200 |
| Rossi | C2 | Milano | | C1 | A1 | 100 |
| … | … | … | | … | … | … |
| Bianchi | C3 | Roma | | C1 | A1 | 100 |
| … | … | … | | … | … | … |
| Verdi | C4 | Roma | | | A3 | 200 |
| … | … | … | | … | … | … |

# Correct solution

| **Name** | **C#** | **Town** | O# | **CC#** | **A#** | **N-pieces** |
|---|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | C1 | A1 | 100 |
| Rossi | C1 | Roma | O1 | C1 | A2 | 200 |
| Rossi | C1 | Roma | O1 | C1 | A3 | 100 |
| Rossi | C2 | Milano | O2 | C2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | C3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | C4 | A3 | 200 |

$$\sigma_{C\#=CC\#}(Customer \times OrderR)$$

# A more elegant solution

| Name | C# | Town | O# | CC# | A# | N-pieces |
|---|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | C1 | A1 | 100 |
| Rossi | C1 | Roma | O1 | C1 | A2 | 200 |
| Rossi | C1 | Roma | O1 | C1 | A3 | 100 |
| Rossi | C2 | Milano | O2 | C2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | C3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | C4 | A3 | 200 |

$$\pi_{\text{Name C\# Town O\# A\# N-pieces}}(\sigma_{\text{C\#=CC\#}}(\text{Customer} \times \textit{OrderR}))$$

we eliminate the duplicate attributes...

# A more elegant solution

| Name | C# | Town | O# | A# | N-pieces |
|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A1 | 100 |
| Rossi | C1 | Roma | O1 | A2 | 200 |
| Rossi | C1 | Roma | O1 | A3 | 100 |
| Rossi | C2 | Milano | O2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | A3 | 200 |

$$\pi_{\text{Name C\# Town O\# A\# N-pieces}}(\sigma_{\text{C\#=CC\#}}(\text{Customer} \times \textit{OrderR}))$$

we eliminate the duplicate attributes...

# A more complicated query

| Name | C# | Town | O# | A# | N-pieces |
|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A1 | 100 |
| Rossi | C1 | Roma | O1 | A2 | 200 |
| Rossi | C1 | Roma | O1 | A3 | 100 |
| Rossi | C2 | Milano | O2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | A3 | 200 |

query: data of the customers and orders of more than 100 pieces

first, we focus on the correct tuples...

# A more complicated query

| Name | C# | Town | O# | A# | N-pieces |
|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A2 | 200 |
| Rossi | C2 | Milano | O2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | A3 | 200 |

query: data of the customers (and orders) who ordered more than 100 pieces

$$\pi_{\text{Name C\# Town O\# A\# N-pieces}}(\sigma_{\text{C\#=CC\#} \wedge \text{ N-pieces>100}}(\text{Customer} \times \textit{OrderR}))$$

# Natural join

$$r_1 \bowtie r_2$$

- it selects the tuples in the result of the Cartesian product that satisfy the condition:

- $R_1.A_1 = R_2.A_1 \wedge R_1.A_2 = R_2.A_2 \wedge \ldots \wedge R_1.A_k = R_2.A_k$

  (where $A_1, A_2, \ldots, A_k$ are the attributes in common between the relations involved in the product)

- duplicate attributes are automatically dropped

# Natural join and cartesian product

$$r_1 \bowtie r_2 = \pi_{XY}(\sigma_C (r_1 \times r_2))$$

where:

- C: $R_1.A_1 = R_2.A_1 \wedge \ldots \wedge R_1.A_k = R_2.A_k$
- X is the set of attributes of $r_1$
- Y is the set of attributes of $r_2$ that are not in $r_1$

important points to remember:

- the attributes in the condition have the same names
- only the tuples having the same values for the attributes in common are merged and returned

# Natural join

**Customer**

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

**Order**

| O# | C# | A# | N-pieces |
|---|---|---|---|
| O1 | C1 | A1 | 100 |
| O2 | C2 | A2 | 200 |
| O3 | C3 | A2 | 150 |
| O4 | C4 | A3 | 200 |
| O1 | C1 | A2 | 200 |
| O1 | C1 | A3 | 100 |

query: data of all the customers and the corresponding orders

$$Customer \bowtie Order$$

# Result

| Name | C# | Town | O# | A# | N-pieces |
|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A1 | 100 |
| Rossi | C1 | Roma | O1 | A2 | 200 |
| Rossi | C1 | Roma | O1 | A3 | 100 |
| Rossi | C2 | Milano | O2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | A3 | 200 |

# Example 1

**query :** names of the customers who ordered more than 100 pieces

**Customer**

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

**Order**

| O# | C# | A# | N-pieces |
|---|---|---|---|
| O1 | C1 | A1 | 100 |
| O2 | C2 | A2 | 200 |
| O3 | C3 | A2 | 150 |
| O4 | C4 | A3 | 200 |
| O1 | C1 | A2 | 200 |
| O1 | C1 | A3 | 100 |

$\pi_{\text{Name}}(\sigma_{\text{N-pieces}>100}(\text{Customer} \bowtie \text{Order}))$

but be careful...

# Example 1

| Name | C# | Town | O# | A# | N-pieces |
|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A1 | 100 |
| Rossi | C1 | Roma | O1 | A2 | 200 |
| Rossi | C1 | Roma | O1 | A3 | 100 |
| Rossi | C2 | Milano | O2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | A3 | 200 |

Customer $\bowtie$ Order

# Example 1

| Name | C# | Town | O# | A# | N-pieces |
|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A2 | 200 |
| Rossi | C2 | Milano | O2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | A3 | 200 |

$$\sigma_{\text{N-pieces}>100}(\text{Customer} \bowtie \text{Order})$$

# Example 1

| Name |
| --- |
| Rossi |
| Bianchi |
| Verdi |

$\pi_{\text{Name}}(\sigma_{\text{N-pieces}>100}(\text{Customer} \bowtie \text{Order}))$

note that the name alone is not a unique identifier of the customer

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

| Name | Town |
| --- | --- |
| Rossi | Roma |
| Rossi | Milano |
| Bianchi | Roma |
| Verdi | Roma |

but, still, you can have duplicates (more than customer having the same name and living in the same city), so it is better to include a key in the projection:

$\pi_{\text{Name},}$

# Example 2

query : names and towns of the customers who ordered more than 100 pieces of articles that cost more than 2 (euros)

**Customer**

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

**Article**

| A# | Label | Price |
|---|---|---|
| A1 | Plate | 3 |
| A2 | Glass | 2 |
| A3 | Mug | 4 |

**Order**

| O# | C# | A# | N-pieces |
|---|---|---|---|
| O1 | C1 | A1 | 100 |
| O2 | C2 | A2 | 200 |
| O3 | C3 | A2 | 150 |
| O4 | C4 | A3 | 200 |
| O1 | C1 | A2 | 200 |
| O1 | C1 | A3 | 100 |

# Example 2

| Name | C# | Town | O# | A# | N-pieces | Label | Price |
|---|---|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A1 | 100 | Plate | 3 |
| Rossi | C1 | Roma | O1 | A2 | 200 | Glass | 2 |
| Rossi | C1 | Roma | O1 | A3 | 100 | Mug | 4 |
| Rossi | C2 | Milano | O2 | A2 | 200 | Glass | 2 |
| Bianchi | C3 | Roma | O3 | A2 | 150 | Glass | 2 |
| Verdi | C4 | Roma | O4 | A3 | 200 | Mug | 4 |

$$(\text{Customer} \bowtie \text{Order}) \bowtie \text{Article}$$

Also in this case we are interested in a meaningful subset of the Cartesian product, that is, the one in which we combine only the information about the objects that are **really associated**

# Example 2

| Name | C# | Town | O# | A# | N-pieces | Label | Price |
|---|---|---|---|---|---|---|---|
| Verdi | C4 | Roma | O4 | A3 | 200 | Mug | 4 |

$$\sigma_{\text{N-pieces}>100 \wedge \text{Price}>2}((\text{Customer} \bowtie \text{Order}) \bowtie \text{Article})$$

# Example 2

| Name | Town |
| --- | --- |
| Verdi | Roma |

$$\pi_{\text{Name,Town}}(\sigma_{\text{N-pieces}>100 \wedge \text{Price}>2}((\text{Customer} \bowtie \text{Order}) \bowtie \text{Article}))$$

# Example 2: alternative solution

query: names and towns of the customers who ordered more than 100 pieces of articles that cost more than 2 (euros)

**Customer**

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

**Order**

| O# | C# | A# | N-pieces |
|---|---|---|---|
| O1 | C1 | A1 | 100 |
| O2 | C2 | A2 | 200 |
| O3 | C3 | A2 | 150 |
| O4 | C4 | A3 | 200 |
| O1 | C1 | A2 | 200 |
| O1 | C1 | A3 | 100 |

**Article**

| A# | Label | Price |
|---|---|---|
| A1 | Plate | 3 |
| A2 | Glass | 2 |
| A3 | Mug | 4 |

by **initially** selecting the tuples we are interested in, the operation is more efficient, as we are avoiding to process useless data

# Example 2: alternative solution

| O# | C# | A# | N-pieces |
|---|---|---|---|
| O2 | C2 | A2 | 200 |
| O3 | C3 | A2 | 150 |
| O4 | C4 | A3 | 200 |
| O1 | C1 | A2 | 200 |

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

# Example 2: alternative solution

| Name | C# | Town | O# | A# | N-pieces |
|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A2 | 200 |
| Rossi | C2 | Milano | O2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | A3 | 200 |

$$\text{Customer} \bowtie \sigma_{\text{N-pieces}>100}(\text{Order})$$

# Example 2 : alternative solution

**Article**

| A# | Label | Price |
|---|---|---|
| A1 | Plate | 3 |
| A2 | Glass | 2 |
| A3 | Mug | 4 |

$\pi_{A\#,Price}$

| A# | Price |
|---|---|
| A1 | 3 |
| A2 | 2 |
| A3 | 4 |

$\pi_{A\#,Price}(Article)$

$\sigma_{Price>2}$ ↓

| A# | Price |
|---|---|
| A1 | 3 |
| A3 | 4 |

$\sigma_{Price>2}(\pi_{A\#,Price}(Article))$

# Example 2 : alternative solution

| Name | C# | Town | O# | A# | N-pieces |
|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A2 | 200 |
| Rossi | C2 | Milano | O2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | A3 | 200 |

$\bowtie$

| A# | Price |
|---|---|
| A1 | 3 |
| A3 | 4 |

$\sigma_{Price>2}(\pi_{A\#,Price}(Article))$

$Customer \bowtie \sigma_{N\text{-}pieces>100}(Order)$

| Name | C# | Town | A# | N-pieces | Price |
|---|---|---|---|---|---|
| Verdi | C4 | Roma | A3 | 200 | 4 |

$(Customer \bowtie \sigma_{N\text{-}pieces>100}(Order)) \bowtie \sigma_{Price>2}(\pi_{A\#,Price}(Article))$

# Example 2

| Name | Town |
| --- | --- |
| Verdi | Roma |

$$\pi_{\text{Name,Town}}((\text{Customer} \bowtie \sigma_{\text{N-pieces}>100}(\text{Order})) \bowtie \sigma_{\text{Price}>2}(\pi_{\text{A\#,Price}}(\text{Article})))$$

# Natural join: special cases

special case 1:

- the 2 relations have some attributes in common but the attributes have no values in common
- result: **empty set!**

query: names and towns of the customers who ordered more than 100 pieces of articles that cost **less** than 2 (euros)

**Article (new version)**

| A# | Label | Price |
|---|---|---|
| A1 | Plate | 3 |
| A2 | Glass | 2 |
| A3 | Mug | 4 |
| A4 | Piattino | 1 |

| A# | Price |
|---|---|
| A4 | 1 |

$\sigma_{Price<2}(\pi_{A\#,Price}(Article))$

# Natural join: special cases

query: names and towns of the customers who ordered more than 100 pieces of articles that cost **less** than 2 (euros)

the join between Customer **and Order** is the same as before, but:

| Name | C# | Town | O# | A# | N-pieces |
|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | A2 | 200 |
| Rossi | C2 | Milano | O2 | A2 | 200 |
| Bianchi | C3 | Roma | O3 | A2 | 150 |
| Verdi | C4 | Roma | O4 | A3 | 200 |

$\bowtie$

| A# | Price |
|---|---|
| A4 | 1 |

$\sigma_{Price<2}(\pi_{A\#,Price}(Article))$

$Customer \bowtie \sigma_{N\text{-}pieces>100}(Order)$

… there are no tuples with the same value for attribute A#

# Natural join: special cases

special case 2:

- the two relations **do not** have attributes with the same name, so the condition

$$R_1.A_1 = R_2.A_1 \wedge R_1.A_2 = R_2.A_2 \wedge \ldots \wedge R_1.A_k = R_2.A_k$$

cannot be evaluated and the natural join degenerates into the Cartesian product

# Natural join: special cases

**Customer**

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

**Order**

| O# | CC# | A# | N-pieces |
|---|---|---|---|
| O1 | C1 | A1 | 100 |
| O2 | C2 | A2 | 200 |
| O3 | C3 | A2 | 150 |
| O4 | C4 | A3 | 200 |
| O1 | C1 | A2 | 200 |
| O1 | C1 | A3 | 100 |

# Natural join: special cases

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

| Name | C# | Town | O# | CC# | A# | N-pieces |
|---|---|---|---|---|---|---|
| Rossi | C1 | Roma | O1 | C1 | A1 | 100 |
| Rossi | C1 | Roma | O2 | C2 | A2 | 200 |
| Rossi | C1 | Roma | O3 | C3 | A2 | 150 |
| Rossi | C1 | Roma | O4 | C4 | A3 | 200 |
| Rossi | C1 | Roma | O1 | C1 | A2 | 200 |
| Rossi | C2 | Milano | O1 | C1 | A1 | 100 |
| … | … | … | | … | … | … |
| Bianchi | C3 | Roma | O3 | C1 | A1 | 100 |
| … | … | … | | … | … | … |
| Verdi | C4 | Roma | O4 | | A3 | 200 |
| … | … | … | | … | … | … |

# Natural join: special cases

- solution:

$$\text{OrderR} = \rho_{C\# \to CC\#}(\text{Order})$$

# Natural join: possible errors



- of course, attributes with the same name also have to share the same meaning:

**Artist**

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

**Painting**

| Title | C# | Artist |
|---|---|---|
| Title1 | C1 | C1 |
| Title2 | C2 | C3 |
| Title3 | C3 | C1 |
| Title4 | C4 | C2 |
| Title5 | C5 | C4 |
| Title6 | C6 | C2 |

to be correct, the join has to be performed on the attributes Artist.C# and Painting.Artist, so, either we rename those attributes or we use a $\theta$-join

# θ-join

- it selects the tuples resulting from the Cartesian product and satisfying the following condition:
  - AθB

where:

- θ is a comparison operator ($\theta \in \{<, =, >, \leq, \geq\}$),
- A is an attribute of the first relation,
- B is an attribute of the second relation
- dom(A)=dom(B)

$$r_1 \bowtie r_2 = \sigma_{A\theta B}(r_1 \times r_2)$$

# Negative conditions

**Customer**

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

query: customers whose name is 'Rossi' and who **do not** live in Roma

| | | |
|---|---|---|
| Rossi | C2 | Milano |

$$\sigma_{\neg(\text{Town}=\text{`Roma'})\wedge\text{Name}=\text{`Rossi'}}(\text{Customer})$$

# Summary

- when we need information from different relations:
  - we identify the needed relations
  - we possibly select subsets of their attributes, and we rename them, if needed
  - we combine the information from different relations using the natural or theta join