Data Management and Analysis

Giuseppe Perelli
perelli@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

The content of the slides is authored by Prof. Maurizio Mancini and 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

What is relational algebra?

- relational algebra is a notation for specifying queries about the contents of relations
- relational algebra eases the task of reasoning about queries
- operations in relational algebra have counterparts in SQL
- to process a query, a DBMS translates SQL into a notation similar to relational algebra

Relational algebra

- formal language to interrogate a relational database, consisting of a set of unary and binary operators that, if applied to one or two relation instances, generate a new relation instance

- procedural language: it describes the exact order in which the operators have to applied to the relation instances to obtain the desired result

What is an algebra?

- an algebra is a structure with a domain and a list of operators
  - arithmetic: domain is a number set, operators are sum, product, etc.
  - set algebra: domains are sets and operators are union, intersection, difference
- an algebra allows us to construct expressions by combining operands and expression using operators and has rules for reasoning about expressions.
  - $a2 + 2 \times a \times b + b2, (a + b)2$
  - $R - (R - S), R \cap S$

Basics of relational algebra

- domains are relations, regarded as sets of tuples
- results of operations are also sets of tuples
- think of expressions in relational algebra as queries, which construct new relations from given ones
- four types of operators:
  - remove parts of a single relation: projection and selection
  - usual set operations: union, intersection, difference
  - combine the tuples of two relations: cartesian product and joins
  - renaming

Projection

- it performs a "vertical cut" on a relation, that is, it selects a subset of the relation's attributes
- represented by the symbol π:

$$\pi_{A_1,A_2,\ldots,A_k}(R)$$

- select the columns of r corresponding to the attributes $A_1,A_2,\ldots,A_k$

Projection

Customer

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

query: get all the customers' names

π_{Name} (Customer) →

Rossi
Bianchi
Verdi

Projection

Customer

Name	C#	Town	……..
Rossi	C1	Roma	……..
Rossi	C2	Milano	……..
Rossi	C3	Roma	……..
Bianchi	C4	Roma	……..
Verdi	C5	Roma	……..

query: get all the customers' names

note: we follow the set rules, so there cannot exist duplicates in the result of the query

π_{Name}(Customer)

Rossi
Bianchi
Verdi

tuple duplicate = a tuple with the exact same values of another tuple

Projection

Customer

Name	**C#**	**Town**	**……..**
Rossi	C1	Roma	……..
Rossi	C2	Milano	……..
Rossi	C3	Roma	……..
Bianchi	C4	Roma	……..
Verdi	C5	Roma	……..

query: find all the customers' names and towns

Rossi	Roma
Rossi	Milano
Bianchi	Roma
Verdi	Milano

$\pi_{Name,Town}(Customer)$

better, but there are still duplicates in the result, so they appear just once

Projection

Customer

Name	C#	Town	……..
Rossi	C1	Roma	……..
Rossi	C2	Milano	……..
Rossi	C3	Roma	……..
Bianchi	C4	Roma	……..
Verdi	C5	Roma	……..

query: get all the customers' names and codes

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

Rossi	C1
Rossi	C2
Rossi	C3
Bianchi	C4
Verdi	C5

to obtain all the Customers, we include the Customer's code C#, which is unique

Selection

- it performs a "horizontal cut" on the relation, that is, it selects all the rows that meet some constraints

- indicated by the symbol sigma σ:

$$\sigma_C(R)$$

- select the tuples of r that meet condition C

Selection

- condition is a composite Boolean expression (using operators $\wedge, \vee$ and $\neg$), whose terms are in the form:

$$A \; \theta \; B$$

or

$$A \; \theta \; \text{'a'}$$

- where:
- θ is a comparison operator ($\theta \in \{<, =, >, \leq, \geq\}$)
- A and B are attributes of the same domain (dom(A)=dom(B))
- a is an item in dom(A) ($a \in dom(A)$) (a constant, or an expression)

Selection

Customer

Name	C#	Town	…….
Rossi	C1	Roma	…….
Rossi	C2	Milano	…….
Rossi	C3	Roma	…….
Bianchi	C4	Roma	…….
Verdi	C5	Roma	…….

query: get full information about the customers living in Roma

$\sigma_{\text{Town='Roma'}}(\text{Customer})$

Rossi	C1	Roma	……..
Rossi	C3	Roma	……..
Bianchi	C4	Roma	……..
Verdi	C5	Roma	……..

Selection

Customer

Name	C#	Town	……..
Rossi	C1	Roma	……..
Rossi	C2	Milano	……..
Rossi	C3	Roma	……..
Bianchi	C4	Roma	……..
Verdi	C5	Roma	……..

query: customers whose name is "Rossi" and who live in Rome

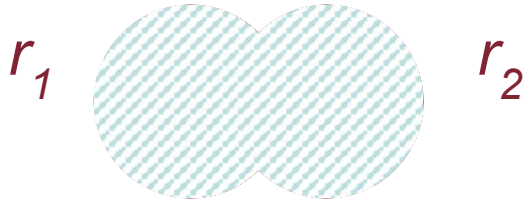
$\sigma_{Town='Roma' \wedge Name='Rossi'}(Customer)$

Rossi	C1	Roma	……..
Rossi	C3	Roma	……..

using the selection operator, there cannot be any loss of tuples...

Union

- creates a new relation instance containing all the tuples belonging to at least one of the operand instances
- it is represented by the symbol ∪



Union

- it can be performed on union compatible operands only
- union compatible operands:
  - have the same number of attributes
  - corresponding attributes have the same domains
- note: attributes do not need to have the same names, even if only in that case the result of the operation makes sense!
  - e.g.: "number of exams" and "age" are defined on the same domain (positive integers) but it would not make any sense to compute the union of a set of relations including those columns (one column each)
  - of course, the operands can be the result of the computation of other operators, for example the projection operator (to eliminate incompatible attributes)

Union

Teachers

Name	Code	Department
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English

Admins

Name	Code	Department
Esposito	C1	English
Riccio	C2	Math
Pierro	C3	Italian
Bianchi	C4	English

Union

AllStaff

Name	Code	Department
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English
Esposito	C1	English
Riccio	C2	Math
Pierro	C3	Italian
Bianchi	C4	English

AllStaff = Teachers ∪ Admins
but... does it make sense?

Union

Teachers

Name	Code	Department
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English

Admins

Name	Code	Department
Esposito	C1	English
Riccio	C2	Math
Pierro	C3	Italian
Verdi	C4	English
Bianchi	C5	English

Union

AllStaff

Name	Code	Department
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English
Esposito	C1	English
Riccio	C2	Math
Pierro	C3	Italian
Bianchi	C5	English

AllStaff = Teachers ∪ Admins
Anyone missing?

Union

Teachers

Name	Code	Department
Rossi	D1	Math
Rossi	D2	Italian
Bianchi	D3	Math
Verdi	D4	English

Admins

Name	Code	Department
Esposito	A1	English
Riccio	A2	Math
Pierro	A3	Italian
Verdi	A4	English
Bianchi	A5	English

Union

AllStaff

Name	Code	Department
Rossi	D1	Math
Rossi	D2	Italian
Bianchi	D3	Math
Verdi	D4	English
Esposito	A1	English
Riccio	A2	Math
Pierro	A3	Italian
Verdi	A4	English
Bianchi	A5	English

AllStaff = Teachers ∪ Admins

Union

Teachers

Name	Code	Department
Rossi	D1	Math
Rossi	D2	Italian
Bianchi	D3	Math
Verdi	D4	English

Admins

Name	Code	Department	Salary
Esposito	A1	English	1250
Riccio	A2	Math	2000
Pierro	A3	Italian	1000

Union

in the previous example the two relations are not union compatible, as they have different attributes

Union

solution: use the projection first, to make them union compatible

AllStaff =

$$\pi_{\text{Name,Code, Department}}(\text{Teachers}) \cup \pi_{\text{Name,Code, Department}}(\text{Admins})$$

Union

Teachers

Name	TCode	Department
Rossi	D1	Math
Rossi	D2	Italian
Bianchi	D3	Math
Verdi	D4	English

Admins

Name	ACode	ServiceY
Esposito	A1	10
Riccio	A2	15
Pierro	A3	2
Bianchi	A4	12

Union

the two relations are not union compatible, as their attributes are defined on different domains

Union

solution: use projection and select the attributes with the same meaning

AllStaff =

$\pi_{Name,TCode}$ (Teachers) $\cup$ $\pi_{Name,ACode}$ (Admins)

Union

Teachers

Name	TCode	Department
Rossi	D1	Math
Rossi	D2	Italian
Bianchi	D3	Math
Verdi	D4	English

Admins

Name	ACode	Office
Esposito	A1	Salary
Riccio	A2	Didactics
Pierro	A3	Administration
Bianchi	A4	Didactics

Union

in this case, the relations are union compatible but their attributes have different meanings

Union

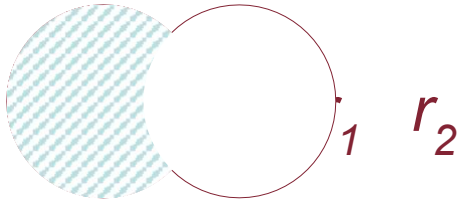
solution: use projection to select the attributes with the same meaning

AllStaff =

$\pi_{Name,TCode}$ (Teachers) $\cup$ $\pi_{Name,ACode}$ (Admins)

Difference

- It can be applied to union compatible operands only
- it creates a relation containing the tuples in the first operand relation but not in the second operand relation
- we use the symbol −



$r_1 - r_2$

Difference

Students

Name	TaxCode	Department
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English

Admins

Nome	TaxCode	Department
Esposito	C5	Italian
Riccio	C6	Math
Pierro	C7	English
Bianchi	C3	Math

Difference

- **note: difference is not commutative, like union**
  - Students – Admins = students who are not admins
  - Admins – Students = admins who are not students

Difference

Students – Admins

Students

Name	TaxCode	Department
Rossi	C1	Math
Rossi	C2	Italian
Verdi	C4	English

Admins - Students

Admins

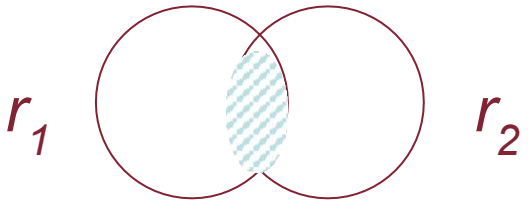
Name	TaxCode	Department
Esposito	C5	Italian
Riccio	C6	Math
Pierro	C7	English

Difference

similarly to union, non-union compatible relations can be projected on subsets of their attributes to allow computing difference

Intersection

- it can be applied to union compatible operands only
- it creates a relation containing the items that are present in both the operand relations
- we use the symbol ∩



- $r_1 \cap r_2 = (r_1 - (r_1 - r_2))$

Intersection

Students

Name	TaxCode	Department
Rossi	C1	Math
Rossi	C2	Italian
Bianchi	C3	Math
Verdi	C4	English

Admins

Name	TaxCode	Department
Esposito	C5	Italian
Riccio	C6	Math
Pierro	C7	English
Bianchi	C3	Math

Intersection

- it is a commutative operand
  - Students ∩ Admins = students who are also admins

StudAdmins

Name	TaxCode	Department
Bianchi	C3	Math

Intersection

similarly to union, non-union compatible relations can be projected on subsets of their attributes to allow computing intersection