Relational Algebra

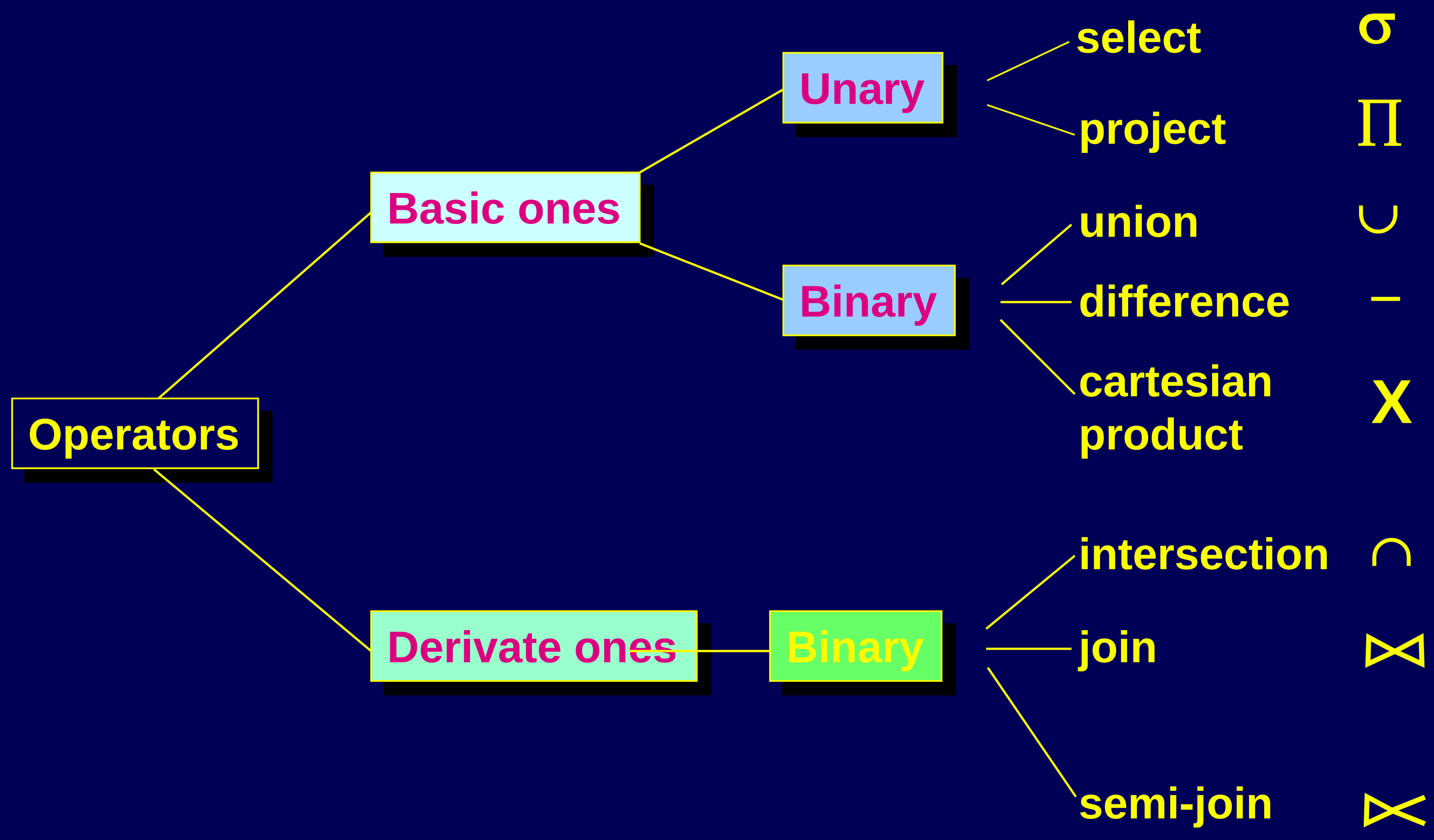
Taxonomy of Languages

- **Formal Languages:**
  - Relational algebra;
  - Relational calculus;
  - Logic programming.
- **Programming languages:**
  - SQL: Structured Query Language;
  - QBE: Query By Example.

Relational Algebra

- Codd (70)
- Useful to learn how to make queries
- Minimal set of 5 operators expressing the entire processing power of the language

Global View


Operators
Basic ones
Derivate ones
Unary
Binary
Binary
select σ
project Π
union ∪
difference −
cartesian product X
intersection ∩
join ⋈
semi-join ⋉

Example: University Exams

student

Id	Name	City	Dept
123	Carlo	Bologna	CS
415	Paola	Torino	CS
702	Antonio	Roma	Log

exam

Id	Course Id	Date	Mark
123	1	7-9-03	10
123	2	8-1-03	8
702	2	7-9-03	5

course

Course Id	Title	Teacher
1	Math	Barozzi
2	CS	Meo

Selection

$\sigma_{\text{Name='Paola'}}$ STUDENT

- Yields to a relation (with no name) where
- schema:
  - Same schema as STUDENT
- instance:
  - Those tuples of STUDENT fulfilling the selection predicate

Id	Name	City	Dept
415	Paola	Torino	CS

Selection Predicate: Syntax

Boolean expression of simple predicates

Boolean expression :

- AND (P1 AND P2) ($\wedge$)
- OR (P1 OR P2) ($\vee$)
- NOT (P1) ($\neg$)

Simple predicates:

- TRUE, FALSE
- term comparator term

comparator:

- =, !=, <, <=, >, >=

term:

- constant, attribute
- Arithmetic expression of terms and attributes

Example

σ STUDENT

(City='Torino') OR

((City='Roma')

AND NOT (Dept=’Log'))

Id	Name	City	Dept
~~123~~	~~Carlo~~	~~Bologna~~	~~CS~~
415	Paola	Torino	CS
~~702~~	~~Antonio~~	~~Roma~~	~~Log~~

Projection

$\Pi_{\text{Name,Dept}}$ STUDENT

- Yields to a relation (with no name) where
- schema:
  - attributes Name and Dept
- instance:
  - the restriction of tuples
  - over the attributes
  - Name and Dept

Name	Dept
Carlo	CS
Paola	CS
Antonio	Log

Projection and Duplicates

- **In the formal model, the projection eliminates the duplicates**

$$\Pi_{\text{Dept}}\ \text{STUDENT}$$

Dept
CS
Log

- **In the informal model (real systems), the elimination of duplicates MUST be explicitly requested (SQL: distinct).**

Assignment

- **Provides the resulting relations with a name**
- **Does not belong to algebraic operators**

$$\textbf{CScientist} = \sigma_{\text{Dept='CS'}}\ \textbf{STUDENT}$$

$$\textbf{Turin} = \sigma_{\text{City='Torino'}}\ \textbf{STUDENT}$$

Union

TABLE1 $\cup$ TABLE2

Can be done iff TABLE1 and TABLE2 are compatible

Thus:
- **Same degree**
- **Or (usual requirement) their domains have the same type – in order**

Union

- CScientist $\cup$ Turin
- Yields to a relation (with no name) where
- schema:
  - Same schema as CScientist
- instance:
  - union of the tuples of CScientist and Turin

Commutative on instances

ID	Name	City	Dept
123	Carlo	Bologna	CS
415	Paola	Torino	CS

Difference

TABLE1 - TABLE2

Can be done iff TABLE1 and TABLE2 are compatible

Difference

- Cscientist - Turin
- Yields to a relation (with no name) where
- schema:
  - Same schema as CScientist
- instance:
  - Difference of the tuples of CScientist and Turin
  - NOT commutative on instances

Id	Name	City	Dept
123	Carlo	Bologna	CS

Cartesian Product

- R × S
- Yields to a relation (with no name) where
- schema:
  - attributes from R and from S
  - degree(RxS)= degree(R)+degree(S)
- instances:
  - All possible pairs of tuples of R and of S
  - card(RxS)=card(R)*card(S)

Example

R1(A,B)

A	B
a	1
b	3

R2(C,D)

C	D
c	1
b	3
a	2

R1xR2 (A,B,C,D)

A	B	C	D
a	1	c	1
a	1	b	3
a	1	a	2
b	3	c	1
b	3	b	3
b	3	a	2

Intersection

- TABLE1 $\cap$ TABLE2

Can be done iff TABLE1 and TABLE2 are compatible

Can be derived by using the next formula:

$$R \cap S = R - (R - S)$$

Intersection

CScientist $\cap$ Touriner

- Yields to a relation (with no name) where
- schema:
  - Same schema as CScientist
- instance:
  - Intersection of tuples of CScientist and Turin
- Commutative on instances

Id	Name	City	Dept
415	Paola	Torino	CS

Join

STUDENT $|\triangleright\triangleleft|_{\text{STUDENT.Id=EXAM.Id}}$ EXAM

Same as:

$\sigma_{\text{STUDENT.Id=EXAM.Id}}$ STUDENT × EXAM

Attributes with the same name are dealt with by the “dot notation”:
EXAM.Id, STUDENT.Id

Join

STUDENT $\bowtie_{\text{STUDENT.Id=EXAM.Id}}$ EXAM

- Yields to a relation (with no name) where
- schema:
  - concatenation of the schemata of STUDENT and EXAM
- instances:
  - The tuples of the Cartesian product that fulfill selection predicate:

STUDENT. Id	Name	City	Dept	EXAM. Id	Course Id	Date	Mark
123	Carlo	Bologna	CS	123	1	7-9-03	10
123	Carlo	Bologna	CS	123	2	8-1-03	8
702	Antonio	Roma	Log	702	2	7-9-03	5

Syntax of the JOIN Predicate

Conjunctive expression of simple predicates

ATTR1 comp ATTR2

where ATTR1 belongs to TAB1
ATTR2 belongs to TAB2
comp: =, !=, <, <=, >, >=

EQUI-JOIN:

equality comparison, ONLY

Natural Join

equi-join of all the predicates with the same name (predicates are omitted, repeated join column is omitted)

STUDENT |⊳⊲| EXAM

Id	Name	City	Dept	Course Id	Date	Mark
123	Carlo	Bologna	CS	1	7-9-03	10
123	Carlo	Bologna	CS	2	8-1-03	8
702	Antonio	Roma	Log	2	7-9-03	5

Natural Join of Three Tables

STUDENT |▷◁| EXAM |▷◁| COURSE

Id	Name	City	Dept	Course Id	Date	Mark	Title	Teacher
123	Carlo	Bologna	CS	1	7-9-03	10	math	Barozzi
123	Carlo	Bologna	CS	2	8-1-03	8	CS	Meo
702	Antonio	Roma	Log	2	7-9-03	5	CS	Meo

Semi-join

STUDENT $\ltimes_{\text{STUDENT.Id=EXAM.Id}}$ EXAM

$\Pi_{\text{Attr(Student)}}$ STUDENT
$\bowtie_{\text{STUDENT.Id=EXAM.Idr}}$ EXAM

- Yields to a relation (with no name) where
- schema:
  - schema of STUDENT
- instance:
  - The tuples obtained by projecting on the attributes of STUDENT the join of STUDENT with EXAM

Id	Name	City	Dept
123	Carlo	Bologna	CS
702	Antonio	Roma	Log

Natural Semi-Join

STUDENT |⊳⊲ EXAM =
$\Pi_{Attr(Student)}$ STUDENT |⊳⊲| EXAM

Project over the attributes of STUDENT of the natural join of STUDENT and EXAM

Id	Name	City	Dept
123	Carlo	Bologna	Inf
702	Antonio	Roma	Log

Equivalence of Expressions

- **Which students got a mark of 10 in Math?**

Π_{Name} **(STUDENT $\bowtie$ ($\sigma_{\text{Mark=10}}$ EXAM $\bowtie$ ($\sigma_{\text{Title='math'}}$ COURSE)))**

- **Equivalent to:**

$\Pi_{\text{Name}}\ \sigma_{\text{Mark=10} \wedge \text{Title='math'}}$ **(STUDENT $\bowtie$ EXAM $\bowtie$ COURSE)**

Equivalence of Expressions

- **Antonio's teachers?**

$$\Pi_{\text{Teacher}} (\text{COURSE} \bowtie (\text{EXAM} \bowtie \sigma_{\text{Name = 'Antonio'}}\ \text{STUDENT}))$$

- **Equivalent to:**

$$\Pi_{\text{Teacher}}\ \sigma_{\text{Name = 'Antonio'}} (\text{STUDENT} \bowtie \text{EXAM} \bowtie \text{COURSE})$$

Complex Expressions

- Find the names of students who never got a mark smaller than 8

$$\Pi_{Name}\, STUDENT \bowtie (\Pi_{Id}\, EXAM - \Pi_{Id}\, \sigma_{Mark<8}\, EXAM)$$

- Explanation: at first, find all the Ids of all the students who passed at least ONE examination, then subtract the Ids of those who got 8 or less, then find their names.

Complex Expressions

- **Find the names of all the students who never got less than 8 OR never passed an examination**

$$\Pi_{Name}\ STUDENT \bowtie (\Pi_{Id}\ EXAM - \Pi_{Id}\ \sigma_{Mark<8}\ EXAM)$$
$$\cup$$
$$\Pi_{Name} (\Pi_{Id}\ STUDENT - \Pi_{Id}\ STUDENT \bowtie EXAM)$$

Complex Expressions

- **Find the names of the students which passed "CS" and "math" the same day**

$$\Pi_{\text{Name}}\ \text{STUDENT} \bowtie$$
$$((\text{ EXAM} \bowtie \sigma_{\text{Title='CS'}}\ \text{COURSE})$$
$$\bowtie_{\text{Id = Id} \wedge \text{Date = Date}}$$
$$(\text{EXAM} \bowtie \sigma_{\text{Title='math'}}\ \text{COURSE}))$$

- **Explanation: at first, find the Ids of the students which passed the two examinations the same day, then find their names.**

Complex Expressions

- **Find the last exam for every student**

EXAM

-

(EXAM $\bowtie_{(Id = Id) \land (Date < Date)}$ EXAM)

- **Explanation: at first, find all the exams which are not the last ones (i.e., they are followed by another exam for that student with a subsequent date), then subtract this set to the set of all the exams. The remainder is the set of the last exams.**

More Exercises

(homework)

Complex Expressions

- Find the first exam for every student;
- Find the exam before the last exam for every student;
- For every student and for every exam, find the next exam of that student (NEXT) :
  - NEXT has the following schema:
    - NEXT(Id, CourseId1, Date1, CourseId2, Date2)

Solutions

(optional exercises)

University Exams

student

Id	Name	City	Dept
123	Carlo	Bologna	CS
415	Paola	Torino	CS
702	Antonio	Roma	Log

exam

Id	Course Id	Date	Mark
123	1	7-9-03	10
123	2	8-1-03	8
702	2	7-9-03	5

corso

Course Id	Title	Teacher
1	matematica	Barozzi
2	informatica	Meo

First Exam

- Find the first exam for every student:
  - **Exam1 = Exam;**
  - **Exam2 = Exam;**
  - **FirstExam = Exam**

-

(Exam1 $|\triangleright\triangleleft$ Exam2

(Exam1.Id = Exam2.Id) $\wedge$

(Exam1.Date > Exam2.Date)

Last before the Last

- Find the last exam before the last exam:
  - find all the exams MINUS the last exam for every student;
  - starting from the above result, find the last exam.

Next Exam

- For every student and for every exam, find the next exam (of that student).
- The NEXT relation will have a schema like the following one:
  - NEXT(Id, Course1, Date1, Course2, Date2)

NEXT

- ESAME1 = ESAME;
- ESAME2 = ESAME;
- ESAME3 = ESAME;

NEXT =

ESAME |▷◁| ESAME1

(ESAME.Matr = ESAME1.Matr $\wedge$

ESAME.Data < ESAME1.Data)

\-

$\Pi_{E1,\ E3}$ (ESAME1 |▷◁| ESAME2 |▷◁| ESAME3)

(ESAME1.Matr = ESAME2.Matr $\wedge$

ESAME2.Matr = ESAME3.Matr $\wedge$

ESAME1.Data < ESAME2.Data $\wedge$

ESAME2.Data < ESAME3.Data)

NEXT

- Ovvero:
  - date le tuple (1,2,3,4);
  - la prima parte della query restituisce (1,2), (1,3), (1,4), (2,3), (2,4), (3,4);
  - la seconda parte restitusice come risultato intermedio (1,2,3), (1,2,4), (1,3,4), (2,3,4).
  - la proiezione sugli attributi di Esame1 ed Esame3 ($\Pi_{E1, E3}$) traforma questo risultato intermedio in (1,3), (1,4), (1,4), (2,4);
  - la differenza finale restituisce (1,2), (2,3), (3,4).

… e si vi fossero due o piu' esami sostenuti nel medesimo giorno????