Relational Algebra

Davood Rafiei

(some material from textbooks and other instructors)

Relational Query Languages

- Languages for describing queries on a relational database
- Three variants
  - Relational Algebra
  - Relational Calculus
  - SQL
- Query languages v.s. programming languages
  - QLs not expected to be "Turing complete".
  - QLs support easy, efficient access to large data sets.

SQL & Relational Algebra

- *Structured Query Language* (SQL)
  - Predominant application-level query language
  - Declarative
- *Relational Algebra*
  - Intermediate language used within DBMS
  - Procedural

Algebra

- Study of operations in an abstract level on some domains
  - e.g. operations +, - and * on natural numbers
- As a language, study of syntax and semantics of expressions
  - e.g 2+3, (46–3)+3, (7*x)+(3*x)
- Relational algebra
  - Domain: set of all relations
  - Expressions: referred to as *queries*

Operations on Tables

id	*name*	*address*	*hobby*
1122	John	123-34 Ave	hockey
1133	Joe	125-34 Ave	biking
2232	Bob	7 Whyte Ave	hockey
5678	John	123-34 Ave	stamps

Person

id	*name*	*office*
1133	Joe	333 Ath
5678	John	222 CSC

Employee

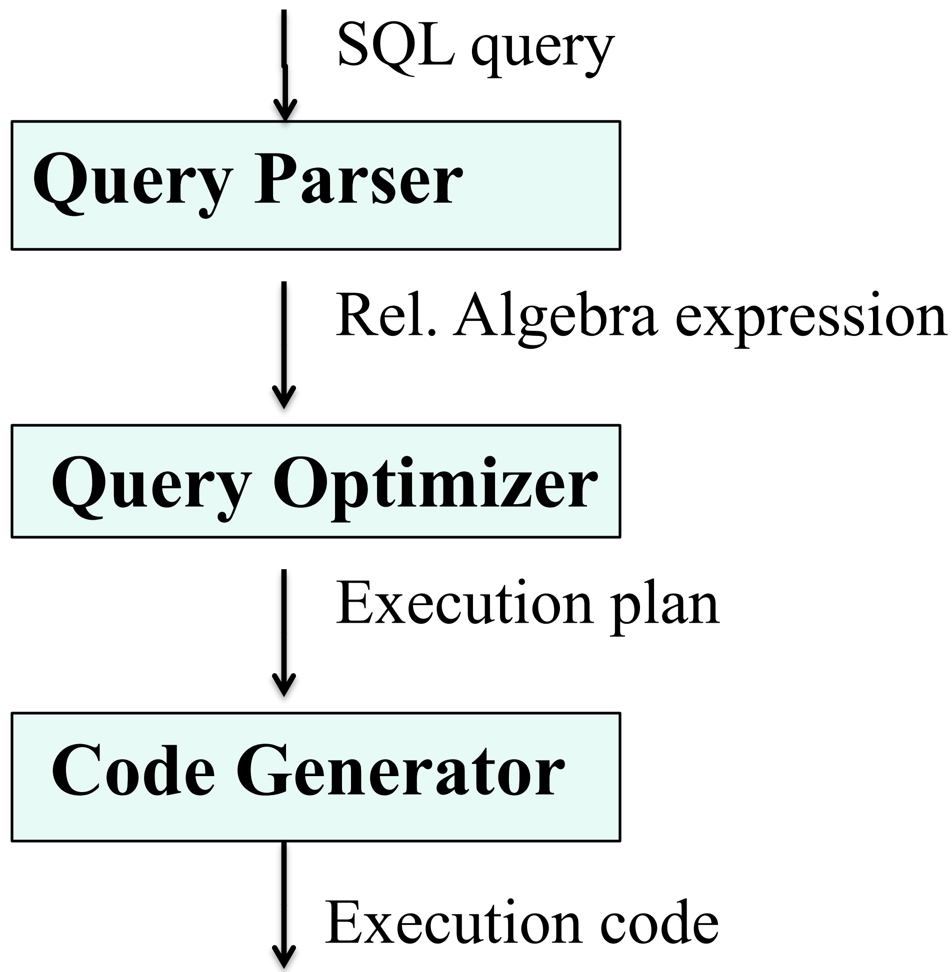
Operations on one table: pick some rows, pick some columns

Operations on two tables: join, …

Relational Algebra

- *Domain*: set of relations
- *Basic operators*: select, project, union, set difference, cartesian product, renaming
- *Derived operators*: set intersection, division, join
- *The language is "procedural"*
  - expressions or queries specify a sequence of operations to be performed to obtain the result

Role inside a DBMS



Select Operator

- Select rows that satisfy the condition

$$\sigma_{condition}\,(relation)$$

- Example:

id	*name*	*address*	*hobby*
1122	John	123-34 Ave	hockey
1133	Joe	125-34 Ave	biking
2232	Bob	7 Whyte Ave	hockey
5678	John	123-34 Ave	stamps

Person

$$\sigma_{hobby=\text{'hockey'}}(\text{Person})$$

id	*name*	*address*	*hobby*
1122	John	123-34 Ave	hockey
2232	Bob	7 Whyte Ave	hockey

Selection Condition

- Operators: $<, \leq, \geq, >, =, \neq$
- Simple selection condition:
  - *<attribute> operator <constant>*
  - *<attribute> operator <attribute>*
- *<condition>* AND *<condition>*
- *<condition>* OR *<condition>*
- NOT *<condition>*

More Examples

- $\sigma_{\, id>1000 \text{ OR } hobby=\text{'hockey'}}$ (Person)
- $\sigma_{\, id>3000 \text{ AND } id<3500}$ (Person)
- $\sigma_{\, \text{NOT}(hobby=\text{'hockey'})}$ (Person)
- $\sigma_{\, hobby\neq\text{'hockey'}}$ (Person)

Project Operator

- Project on (or pick) a subset of columns

$$\pi_{attribute\ list}(relation)$$

- Example:

$$\pi_{name,hobby}(\text{Person})$$

id	*name*	*address*	*hobby*
1122	John	123-34 Ave	hockey
1133	Joe	125-34 Ave	biking
2232	Bob	7 Whyte Ave	hockey
5678	John	123-34 Ave	stamps

Person

name	*hobby*
John	hockey
Joe	biking
Bob	hockey
John	stamps

Example

id	*name*	*address*	*hobby*
1122	John	123-34 Ave	hockey
1133	Joe	125-34 Ave	biking
2232	Bob	7 Whyte Ave	hockey
5678	John	123-34 Ave	stamps

Person

$\pi_{name,address}$(Person)

name	*address*
John	123-34 Ave
Joe	125-34 Ave
Bob	7 Whyte Ave

Result is a relation (meaning no duplicates)!

Expressions

$$\pi_{\,id,\ name}\,(\sigma_{\,hobby=\text{'stamps' OR }hobby=\text{'biking'}}\,(\text{Person}))$$

id	*name*	*address*	*hobby*
1122	John	123-34 Ave	hockey
1133	Joe	125-34 Ave	biking
2232	Bob	7 Whyte Ave	hockey
5678	John	123-34 Ave	stamps

Person

id	*name*
1133	Joe
5678	John

Set Operators

- Relation ~ a set of tuples
  - Set operations apply
- Operations: $\cap$, $\cup$, – (set difference)
- Defined (or meaningful) between relations that have the same structure (called *union compatible relations*)

Union Compatible Relations

- Two relations are *union compatible* if
  - Both have the same number of columns
  - Names of attributes are the same in both
  - Attributes with the same name in both relations have the same domain
- Union compatible relations can be combined using *union*, *intersection*, and *set difference*

Example

Tables:

Person (*SSN, Name, Address, Hobby*)

Professor (*Id, Name, Office, Phone*)

are <u>not</u> union compatible, but

π_{name} (Person) and π_{name} (Professor)

<u>are</u> union compatible and

π_{name} (Person) - π_{name} (Professor)

makes sense.

Cartesian Product

- $R \times S = \{<x,y> \mid x \text{ in } R, y \text{ in } S\}$
  - $R \times S$ is the set of all concatenated tuples $<x,y>$, where x is a tuple in R and y is a tuple in S
  - Relations don't have to be union-compatible
- $R \times S$ can be huge (and expensive to compute)
  - Factor of two in the size of each row
  - Quadratic in the number of rows

a	*b*
a_1	b_1
a_2	b_2

R

c	*d*
c_1	d_1
c_2	d_2

S

a	*b*	*c*	*d*
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_2	b_2	c_1	d_1
a_2	b_2	c_2	d_2

$R \times S$

Renaming

- $\rho_{x(A1,\ A2,\ \ldots\ An)}$ expr
  - *expr* returns a relation
  - Rename the resulting relation to x with the first column in the result relation to A_1, the second to A_2, etc.
- Example
  - Let *R(a,b)* be a relation with two columns
  - $\rho_{(p,\ q,\ r,\ s)} R \times R$ is a relation with 4 columns *p*, *q*, *r* and *s*.
- Common usage
  - To clean up the result
  - To prepare the result for the next operation

Example

Transcript (*sid, cid, sem, grade*)

Teaching (*pid, cid, sem*)

$Transcript \times Teaching$ is not defined, but the following is:

$$\rho_{(sid,\ cid1)}(\pi_{sid,\ cid}(\text{Transcript})) \times \rho_{(pid,\ cid2)}\pi_{pid,\ cid}(\text{Teaching})$$

The result is a relation with 4 attributes:

sid, cid1, pid, cid2

Join (derived operator)

A (*general* or *theta*) *join* of R and S is the expression

$$R \bowtie_{\text{join-condition}} S$$

where *join-condition* is a *conjunction* of terms:

$$A_i \ \text{oper} \ B_i$$

in which A_i is an attribute of R; B_i is an attribute of S; and *oper* is one of $=, <, >, \geq \neq, \leq$.

Equivalent to

$$\sigma_{\text{join-condition}'} (R \times S)$$

where *join-condition* and *join-condition*′ are the same, except for possible renaming of attributes (next)

Join and Renaming

- **Problem**: If *R* and *S* have attributes with the same name, then the Cartesian product is not well-defined
- **Two Solutions**:
  - Rename attributes prior to forming the product and use new names in *join-condition*′.
  - Common attribute names are qualified with relation names in the result of the join

Join – Example

Find employees who earn more than their managers.

Tables: Employee(name, id, salary, mngrId)
Manager(name, id, salary)

$$\pi_{\text{Employee}.name}(\text{Employee} \bowtie_{mngrId=manager.id \text{ AND } Employee.salary>Manager.salary} \text{Manager})$$

The join yields a table with attributes:

Employee.*name*, Employee.*id*, Employee.*salary*, *mngrId*
Manager.*name*, Manager.*id*, Manager.*salary*

Equijoin - Example

Equijoin: Join condition is a conjunction of *equalities*.

$$\pi_{name,cid}(\text{Student} \bowtie_{id=sid} \sigma_{grade=\text{'A'}}(\text{Transcript}))$$

Student

id	*name*	*addr*	*status*
111	John	…..	…..
222	Mary	…..	…..
333	Bill	…..	…..
444	Joe	…..	…..

Transcript

sid	*cid*	*sem*	*grade*
111	114	F10	B
222	115	F10	A
333	201	W10	A

Mary	115
Bill	201

The equijoin is very useful since it combines related data in different relations.

Natural Join

- Special case of equijoin:
  - join condition equates *all* and *only* those attributes with the same name (condition doesn't have to be explicitly stated)
  - duplicate columns eliminated from the result

Transcript (*sid*, *cid*, *sem*, *grade*)
Teaching (*pid*, *cid*, *sem*)

$$\text{Transcript} \bowtie \text{Teaching} =$$

$$\pi_{sid,\ Transcript.cid,\ Transcript.sem,\ grade,\ pid}$$

$$(\ \text{Transcript} \bowtie_{Transcript.cid=Teaching.cid \text{ AND } Transcript.sem=Teaching.sem} \text{Teaching}\ )$$

Natural Join Means

- More generally:

$$R \bowtie S = \pi_{attr\text{-}list} \left(\sigma_{join\text{-}cond} (R \times S) \right)$$

  - *attr-list* = *attributes* (R) $\cup$ *attributes* (S) (duplicates are eliminated)
  - *join-cond* has the form $A_1 = A_1$ AND … AND $A_n = A_n$
  - $\{A_1 \ldots A_n\}$ = *attributes*(R) $\cap$ *attributes*(S)

Example

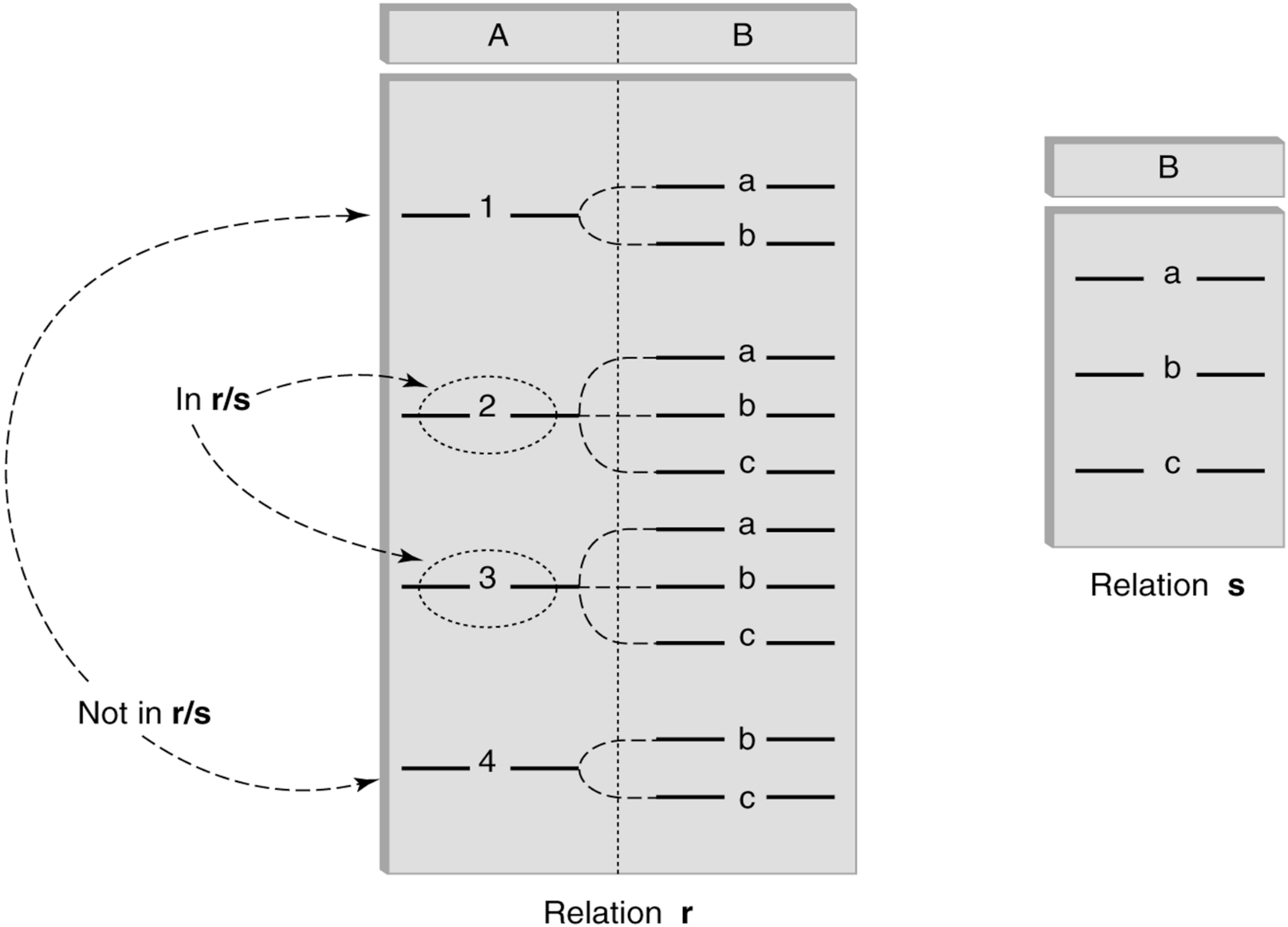
- List the id's of students who took at least two different courses.

  Transcript(*sid, cid, sem, grade*)

- *Transcript* $\bowtie$ *Transcript* won't work!
- Solution?

Division (derived operator)

- Finds tuples in one relation, r, that match *all* tuples in another relation, s
  - $r\ (A_1,\ \ldots A_n,\ B_1,\ \ldots B_m)$
  - $s\ (B_1\ \ldots B_m)$
  - *r/s*, with attributes $A_1, \ldots A_n$, is the set of all tuples $<a>$ such that for every tuple $<b>$ in *s*, $<a,b>$ is in *r*
- Can be expressed in terms of projection, set difference, and cross-product



Source: Kifer, Bernstein, Lewis, Database systems, 2005.

Division - Example

- List the Ids of students who have passed *all* courses taught in winter 2010
- *Numerator*:
  - *sid* and *cid* for every course passed by every student:

$$\pi_{sid,\ cid}(\sigma_{Grade \neq \text{'F'}}(\text{Transcript}))$$

- *Denominator*:
  - *cid* of all courses taught in winter 2010

$$\pi_{cid}(\sigma_{Sem = \text{'W10'}}(\text{Teaching}))$$

- Result is *numerator/denominator*