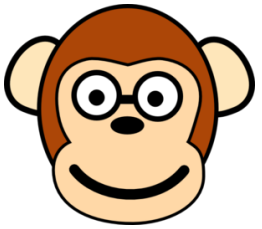
Relational Algebra (Cont.)

csc343, Introduction to Databases
Nosayba El-Sayed (based on slides from Diane Horton)
Fall 2015

UNIVERSITY OF TORONTO

DCS50

Announcements – Week 3



- Assignment #1 will be posted tomorrow (Relational Algebra queries)

- Today:
  - More operations in RA (max/min, at-least, etc)
  - Solving advanced RA examples ;-)

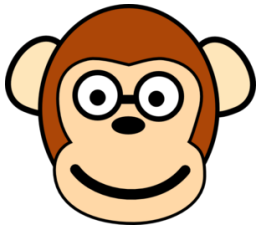
Summary of RA operators

Operation	Name	Symbol
choose rows	select	σ
choose columns	project	π
combine tables	Cartesian product	×
	natural join	⋈
	theta join	$\bowtie_{condition}$
rename relation [and attributes]	rename	ρ
assignment	assignment	:=

Set Operators

Operation	Name	Symbol
get rows that exist in *both* left and right tables	intersection	∩
get rows that exist in left table *or* right table	union	∪
get rows that exist in left table but *not* in right table	difference	–

Cardinality of some operators



- Given R1(X_1) and R2(X_2), the cardinality (number of tuples) of:
  - Cartesian product: |R1 x R2| = |R1| * |R2|
  - Natural join: 0 ≤ | R1 ⋈ R2 | ≤ |R1| * |R2|
  - Notice that *if* the attributes used in a natural join (i.e. the attributes in X1∩X2) contain a key for R2, then:

    0 ≤ | R1 ⋈ R2 | ≤ |R1|
  - Sample applies for left/right outer joins!

E.g. Last week's example (scenarios: DeptName is key; DeptName is **not** key)

Employee

Name	EmpId	DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Tim	1123	Executive

Dept

DeptName	Manager
Sales	Harriet
Production	Charles
Sales	Paul

Employee ⋈ Dept

Name	EmpId	DeptName	Manager
Harry	3415	Finance	ω
Sally	2241	Sales	Harriet
George	3401	Finance	ω
Harriet	2202	Sales	Harriet
Tim	1123	Executive	ω
Sally	2241	Sales	Paul
Harriet	2202	Sales	Paul

Specific types of query

- **Max** (min is analogous):
  - Not directly supported in relational algebra.
  - **Idea**
    - Pair tuples and find those that are *not* the max.
    - Then subtract from all to find the maxes!

max(GPA)?

S

StudentID	GPA
100029	3.25
100210	3.62
202020	2.78

$\rho_{S1}(S) \times \rho_{S2}(S)$

S1.StudentID	S1.GPA	S2.StudentID	S2.GPA
~~100029~~	~~3.25~~	~~100029~~	~~3.25~~
✓ 100029	3.25	100210	3.62
~~100029~~	~~3.25~~	~~202020~~	~~2.78~~
~~100210~~	~~3.62~~	~~100029~~	~~3.25~~
~~100210~~	~~3.62~~	~~100210~~	~~3.62~~
~~100210~~	~~3.62~~	~~202020~~	~~2.78~~
✓ 202020	2.78	100029	3.25
✓ 202020	2.78	100210	3.62
~~202020~~	~~2.78~~	~~202020~~	~~2.78~~

$S3 := \sigma_{(S1.GPA < S2.GPA) \cdots}$?

$\pi_{S3.GPA}$? {3.25, 2.78}

$\pi_{S.GPA} - \pi_{S3.GPA}$?

{3.62}

Specific types of query

- "k or more":
  - Make all combos of *k* different tuples that satisfy the condition.

Students with **two or more** classes with grade > 80?

R

sID	Class	Grade
100029	csc207-2015H	75
100029	csc343-2015S	81
100029	csc343-2015S	82

$\rho_{R1}(R) \times \rho_{R2}(R)$

R1.sID	R1.Class	R1.Grade	R2.sID	R2.Class	R2.Grade
~~100029~~	~~csc207-2015H~~	~~75~~	~~100029~~	~~csc207-2015H~~	~~75~~
100029	csc207-2015H	75	100029	csc343-2015S	81
100029	csc207-2015H	75	100029	csc343-2015S	82
100029	csc343-2015S	81	100029	csc207-2015H	75
~~100029~~	~~csc343-2015S~~	~~81~~	~~100029~~	~~csc343-2015S~~	~~81~~
100029	csc343-2015S	81	100029	csc343-2015S	82
100029	csc343-2015S	82	100029	csc207-2015H	75
100029	csc343-2015S	82	100029	csc343-2015S	81
~~100029~~	~~csc343-2015S~~	~~82~~	~~100029~~	~~csc343-2015S~~	~~82~~

R3(sID1,C1,G1,sID2,C2,G2):=

$\sigma_{(R1.sID=R2.SID \; \wedge R1.Class \neq R2.Class)}$ .....

Students2OrMore80s :=

$\pi_{R3.sID1} \sigma_{(R3.G1>80 \; \wedge R3.G2>80)}$

{100029}

Specific types of query

- Max (min is analogous):
  - Not directly supported in relational algebra
  - Pair tuples and find those that are *not* the max.
  - Then subtract from all to find the maxes.
- "k or more":
  - Make all combos of *k* different tuples that satisfy the condition.
- "exactly k":
  - ["k or more" – "(k+1) or more"]
- "every"..?

Specific types of query

- "every"..
  - Examples:
    - Students with Grade "100" for every course they took (?!)

One possible approach:

- Make all combos that *should* have occurred.

  (e.g. think *Cartesian product* ;-))

- Subtract those that did occur to find those that didn't always.

  ➔ These remaining are the failures.

- Subtract the failures from *all* to get the answer.

Specific types of query

- Max (min is analogous):
  - Not directly supported in relational algebra
  - Pair tuples and find those that are *not* the max.
  - Then subtract from all to find the maxes.
- "k or more":
  - Make all combos of *k* different tuples that satisfy the condition.
- "exactly k":
  - ["k or more" – "(k+1) or more"]
- "every"..?
  - Make all combos that *should* have occurred.
  - Subtract those that *did* occur to find those that didn't always. These remaining are the failures.
  - Subtract the failures from all to get the answer.

Expressing Integrity Constraints

- We've used this notation to expression inclusion dependencies between relations R_1 and R_2:
  $R_1[X] \subseteq R_2[Y]$
- We can use RA to express other kinds of integrity constraints.
- Suppose R and S are expressions in RA. We can write an integrity constraint in either of these ways:
  <RA expression> = ∅
  R ⊆ S (equivalent to saying R - S = ∅)
- We don't need the second form, but it's convenient.

Let’s refresh guidelines for writing queries in relational algebra

Approaching the problem

- Ask yourself which relations need to be involved. Ignore the rest.
- What information are we given? What information is needed?
- Every time you combine relations, confirm that
  - attributes that should match will be ***made*** to match and
  - attributes that will be made to match ***should*** match.

Breaking down the problem

- Remember that you must look one tuple at a time.
  - If you need info from two different tuples, you must make a new relation where it's in one tuple.
- Is there an *intermediate* relation that would help you get the final answer?
  - Draw it out with actual data in it.
- Use assignment to define those intermediate relations.
  - Use good names for the new relations.
  - Name the attributes on the LHS each time, so you don't forget what you have in hand.
  - Add a comment explaining exactly what's in the relation.

– Class Exercises –

(Relational Algebra)

ROSI Schema

Students(sID, surName, campus)

Courses(cID, cName, WR)

Offerings(oID, cID, term, instructor)

Took(sID, oID, grade)

Inclusion dependencies:

- Offerings[cID] ⊆ Courses[cID]
- Took[sID] ⊆ Students[sID]
- Took[oID] ⊆ Offerings[oID]