Relational Algebra

csc343, Introduction to Databases
Diane Horton
Winter 2016

UNIVERSITY OF TORONTO

RA Basics
(covered by your week 2 Prep)

Elementary Algebra

- You did algebra in high school
  - $27y^2 + 8y - 3$
- Operands:
- Operators:

Relational Algebra

- Operands: tables
- Operators:
  - choose only the rows you want
  - choose only the columns you want
  - combine tables
  - and a few other things

A schema for our examples

Movies(<u>mID</u>, title, director, year, length)

Artists(<u>aID</u>, aName, nationality)

Roles(<u>mID, aID, character</u>)

Foreign key constraints:

- Roles[mID] $\subseteq$ Movies[mID]
- Roles[aID] $\subseteq$ Artists[aID]

Select: choose rows

- Notation: $\sigma_c(R)$
  - R is a table.
  - Condition c is a boolean expression.
  - It can use comparison operators and boolean operators
  - The operands are either constants or attributes of R.
- The result is a relation
  - with the same schema as the operand
  - but with only the tuples that satisfy the condition

Exercise

- Write queries to find:
  - All British actors
  - All movies from the 1970s
- What if we only want the names of all British actors?
  We need a way to pare down the columns.

Project: choose columns

- Notation: $\pi_{L}(R)$
  - R is a table.
  - L is a subset (not necessarily a proper subset) of the attributes of R.
- The result is a relation
  - with all the tuples from R
  - but with only the attributes in L, and in that order

About project

- Why is it called “project”?
- What is the value of π_{director} (Movies)?
- Exercise: Write an RA expression to find the names of all directors of movies from the 1970s

- Now, suppose you want the names of all characters in movies from the 1970s.
- We need to be able to combine tables.

Cartesian Product

- Notation: R1 x R2
- The result is a relation with
  - every combination of a tuple from R1 concatenated to a tuple from R2
- Its schema is every attribute from R followed by every attribute of S, in order
- How many tuples are in R1 x R2?
- Example: Movies x Roles
- If an attribute occurs in both relations, it occurs twice in the result (prefixed by relation name)

Continuing on with Relational Algebra

Project and duplicates

- Projecting onto fewer attributes can remove what it was that made two tuples distinct.
- Wherever a project operation might "introduce" duplicates, only one copy of each is kept.
- Example:

People

name	age
Karim	20
Ruth	18
Minh	20
Sofia	19
Jennifer	19
Sasha	20

π_{age} People

age
20
18
19

Example of Cartesian product

profiles:

ID	Name
Oprah	Oprah Winfrey
ginab	Gina Bianchini

follows:

a	b
Oprah	ev
edyson	ginab
ginab	ev

profiles X follows:

ID	Name	a	b
Oprah	Oprah Winfrey	Oprah	ev
Oprah	Oprah Winfrey	edyson	ginab
Oprah	Oprah Winfrey	ginab	ev
ginab	Gina Bianchini	Oprah	ev
ginab	Gina Bianchini	edyson	ginab
ginab	Gina Bianchini	ginab	ev

Composing larger expressions

- Math:
  - The value of any expression is a number.
  - So you can "compose" larger expressions out of smaller ones.
  - There are precedence rules.
  - We can use brackets to override the normal precedence of operators.
- Relational algebra is the same.

Joining two relations

Cartesian product can be inconvenient

- It can introduce nonsense tuples.
- You can get rid of them with selects.
- But this is so highly common, an operation was defined to make it easier: natural join.

Natural Join

- Notation: R ⋈ S
- The result is formed by
  - taking the Cartesian product
  - selecting to ensure equality on attributes that are in both relations (as determined *by name*)
  - projecting to remove duplicate attributes.
- Example:
  Artists ⋈ Roles gets rid of the nonsense tuples.

Examples

- The following examples show what natural join does when the tables have:
  - no attributes in common
  - one attribute in common
  - a different attribute in common
- (Note that we change the attribute names for relation `follows` to set up these scenarios.)

profiles:

ID	Name
Oprah	Oprah Winfrey
ginab	Gina Bianchini

follows:

a	b
Oprah	ev
edyson	ginab
ginab	ev

profiles ⋈ follows:

ID	Name	a	b
Oprah	Oprah Winfrey	Oprah	ev
Oprah	Oprah Winfrey	edyson	ginab
Oprah	Oprah Winfrey	ginab	ev
ginab	Gina Bianchini	Oprah	ev
ginab	Gina Bianchini	edyson	ginab
ginab	Gina Bianchini	ginab	ev

profiles:

ID	Name
Oprah	Oprah Winfrey
ginab	Gina Bianchini

follows:

ID	b
Oprah	ev
edyson	ginab
ginab	ev

profiles ⋈ follows:

ID	Name	ID	b
Oprah	Oprah Winfrey	Oprah	ev
~~Oprah~~	~~Oprah Winfrey~~	~~edyson~~	~~ginab~~
~~Oprah~~	~~Oprah Winfrey~~	~~ginab~~	~~ev~~
~~ginab~~	~~Gina Bianchini~~	~~Oprah~~	~~ev~~
~~ginab~~	~~Gina Bianchini~~	~~edyson~~	~~ginab~~
ginab	Gina Bianchini	ginab	ev

(The redundant ID column is omitted in the result)

profiles:

ID	Name
Oprah	Oprah Winfrey
ginab	Gina Bianchini

follows:

a	ID
Oprah	ev
edyson	ginab
ginab	ev

profiles ⋈ follows:

ID	Name	a	ID
~~Oprah~~	~~Oprah Winfrey~~	~~Oprah~~	~~ev~~
~~Oprah~~	~~Oprah Winfrey~~	~~edyson~~	~~ginab~~
~~Oprah~~	~~Oprah Winfrey~~	~~ginab~~	~~ev~~
~~ginab~~	~~Gina Bianchini~~	~~Oprah~~	~~ev~~
ginab	Gina Bianchini	edyson	ginab
~~ginab~~	~~Gina Bianchini~~	~~ginab~~	~~ev~~

(The redundant ID column is omitted in the result)

Properties of Natural Join

- Commutative:

  $$R \bowtie S = S \bowtie R$$

  (although attribute order may vary)
- Associative:

  $$R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$$

- So when writing n-ary joins, brackets are irrelevant. We can just write:

  $$R_1 \bowtie R_2 \bowtie \ \ldots \ \bowtie R_n$$

Questions

1. How many tuples are in Artists × Roles?
2. How many tuples are in Artists ⋈ Roles?
3. What is the result of:

   $$\pi_{\text{aName}}\,\sigma_{\text{director=''Kubrick''}}\,(\text{Artists} \bowtie \text{Roles} \bowtie \text{Movies})$$

4. What is the result of:

   $$\pi_{\text{aName}}\,((\sigma_{\text{director=''Kubrick''}}\,\text{Artists}) \bowtie \text{Roles} \bowtie \text{Movies})$$

UNIVERSITY OF TORONTO

Movies:

mID	title	director	year	length
1	Shining	Kubrick	1980	146
2	Player	Altman	1992	146
3	Chinatown	Polaski	1974	131
4	Repulsion	Polaski	1965	143
5	Star Wars IV	Lucas	1977	126
6	American Graffiti	Lucas	1973	110
7	Full Metal Jacket	Kubrick	1987	156

Artists:

aID	aName	nationality
1	Nicholson	American
2	Ford	American
3	Stone	British
4	Fisher	American

Roles:

mID	aID	nationality
1	1	Jack Torrance
3	1	Jake 'J.J.' Gittes
1	3	Delbert Grady
5	2	Han Solo
6	2	Bob Falfa
5	4	Princess Leia Organa

Special cases for natural join

No tuples match

Employee	Dept
Vista	Sales
Kagani	Production
Tzerpos	Production

Dept	Head
HR	Boutilier

Exactly the same attributes

Artist	Name
9132	William Shatner
8762	Harrison Ford
5555	Patrick Stewart
1868	Angelina Jolie

Artist	Name
1234	Brad Pitt
1868	Angelina Jolie
5555	Patrick Stewart

No attributes in common

Artist	Name
1234	Brad Pitt
1868	Angelina Jolie
5555	Patrick Stewart

mID	Title	Director	Year	Length
1111	Alien	Scott	1979	152
1234	Sting	Hill	1973	130

Natural join can "over-match"

- Natural join bases the matching on attribute names.
- What if two attributes have the same name, but we don't want them to have to match?
- Example: if Artists used "name" for actors' names and Movies used "name" for movies' names.
  - Can rename one of them (we'll see how).
  - Or?

Natural join can "under-match"

- What if two attributes don't have the same name and we *do* want them to match?
- Example: Suppose we want aName and director to match.
- Solution?

Theta Join

- It's common to use σ to check conditions after a Cartesian product.
- Theta Join makes this easier.
- Notation: $R \bowtie_{condition} S$
- The result is
  - the same as Cartesian product (not natural join!) followed by select.
- In other words, $R \bowtie_{condition} S = \sigma_{condition} (R \times S)$.
- The word "theta" has no special connotation. It is an artifact of a definition in an early paper.

- You save just one symbol.
- You still have to write out the conditions, since they are not inferred.

Composing larger expressions (plus a few new operators)

Precedence

- Expressions can be composed recursively.
- Make sure attributes match as you wish.
  - It helps to annotate each subexpression, showing the attributes of its resulting relation.
- Parentheses and precedence rules define the order of evaluation.
- Precedence, from highest to lowest, is:

  σ, π, ρ

  $\times$, $\bowtie$

  $\cap$

  $\cup$, $-$
- Unless very sure, use brackets!

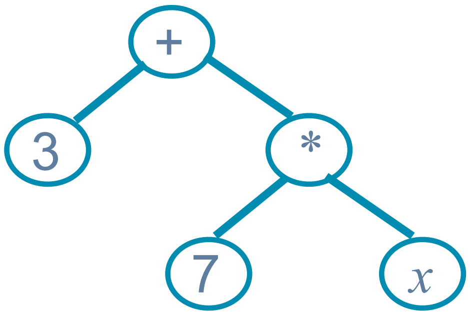
Breaking down expressions

- Complex nested expressions can be hard to read.
- Two alternative notations allow us to break them down:
  - Expression trees.
  - Sequences of assignment statements.

Expression Trees

- Leaves are relations.
- Interior notes are operators.
- Exactly like representing arithmetic expressions as trees.

$3 + 7 * x$



- If interested, see Ullman and Widom, section 2.4.10.

Assignment operator

- Notation:

  $R := \text{Expression}$

- Alternate notation:

  $R(A_1, \ldots, A_n) := \text{Expression}$

  - Lets you name all the attributes of the new relation
  - Sometimes you don't want the name they would get from Expression.

- R must be a temporary variable, not one of the relations in the schema.
  I.e., you are not updating the content of a relation!

- Example:
  - Temp1 := $Q \times R$
  - Temp2 := $\sigma_{a=99}$ (Temp1) $\times$ S
  - Answer(part, price) := $\pi_{b,c}$ (Temp2)
- Whether / how small to break things down is up to you. It's all for readability.
- Assignment helps us break a problem down
- It also allows us to change the names of relations [and attributes].
- There is another way to rename things …

Rename operation

- Notation: $\rho_{R1}(R2)$
- Alternate notation: $\rho_{R1(A1, ..., An)}(R2)$
  - Lets you rename all the attributes as well as the relation.
- Note that these are equivalent:
  $R1(A1, ..., An) := R2$
  $R1 := \rho_{R1(A1, ..., An)}(R2)$
- ρ is useful if you want to rename *within* an expression.

Summary of operators

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

"Syntactic sugar"

- Some operations are not *necessary*.
  - You can get the same effect using a combination of other operations.
- Examples: natural join, theta join.
- We call this "syntactic sugar".
- This concept also comes up in logic and programming languages.

Practise writing queries

Set operations

- Because relations are sets, we can use set intersection, union and difference.
- But only if the operands are relations over the same attributes (in number, name, and order).
- If the names or order mismatch?

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:

  $R = \emptyset$

  $R \subseteq S$ (equivalent to saying $R - S = \emptyset$)
- We don't need the second form, but it's convenient.

Integrity Constraints: Example

- Express the following constraints using the notation $R = \emptyset$ or $R \subseteq S$:

1. 400-level courses cannot count for breadth.
2. In terms when csc490 is offered, csc454 must also be offered.

Summary of techniques for writing queries in relational algebra

Approaching the problem

- Ask yourself which relations need to be involved. Ignore the rest.
- 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.
- Annotate each subexpression, to show the attributes of its resulting relation.

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.

Specific types of query

- Max (min is analogous):
  - 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 are the failures.
  - Subtract the failures from all to get the answer.

Relational algebra wrap-up

RA is procedural

- An RA query itself suggests a procedure for constructing the result
  (*i.e.*, how one could implement the query).
- We say that it is “procedural.”

Evaluating queries

- Any problem has multiple RA solutions.
  - Each solution suggests a "query execution plan".
  - Some may seem a more efficient.
- But in RA, we won't care about efficiency; it's an algebra.
- In a DBMS, queries actually are executed, and efficiency matters.
  - Which query execution plan is most efficient depends on the data in the database and what indices you have.
  - Fortunately, the DBMS optimizes our queries.
  - We can focus on what we want, not how to get it.

Relational Calculus

- Another abstract query language for the relational model.
- Based on first-order logic.
- RC is “declarative”: the query describes what you want, but not how to get it.
- Queries look like this:
  $\{ t \mid t \in \text{Movies} \wedge t[\text{director}] = \text{“Scott”} \}$
- Expressive power (when limited to queries that generate finite results) is the same as RA. It is “relationally complete.”