Relational Algebra

Review and Practice

Recall

- While learning relational algebra, we will assume:
  - Relations are sets, so no two rows are the same.
  - Every cell has a value (no nulls).
  - This is the pure relational model.
- In SQL, we will drop these assumptions.
- But for now, they simplify our queries.
- And we will still build a great foundation for SQL.

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

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

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
  - may need to remove duplicate tuples to ensure result is a **relation**

Project and duplicates

- Projecting onto fewer attributes can remove what it was that made two tuples distinct.
- Wherever a project operation introduces 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

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
- If an attribute occurs in both relations, it occurs twice in the result (prefixed by relation name)

Cartesian product can be inconvenient

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

Natural Join

- Notation: $R \bowtie S$
- The result is defined 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 $\bowtie$ 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)

Properties of (Natural) Join

- Commutative:

  $R \bowtie S = S \bowtie R$

  (although attribute order may vary; this will matter later when we use set operations)

- 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

For the instance on our first (movies) worksheet:

1. How many tuples are in Artists × Roles?
2. How many tuples are in Artists $\bowtie$ 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:

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

1. How many tuples are in Artists × Roles?

Artists:

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

Roles:

mID	aID	character
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

2. How many tuples are in Artists $\bowtie$ Roles?

Artists:

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

Roles:

mID	aID	character
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

3. What is the result of:

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

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	character
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

4. What is the result of:

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

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	character
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

Third Practice Sheet

- ☐ Student(sID, surName, firstName, campus, email, gpa)
- ☐ Course(dept, cNum, name, breadth)

- ☐ Offering(oID, dept, cNum, term, instructor)

- ☐ Took(sID, oID, grade)

- ☐ Offering[dept, cNum] ⊆ Course[dept, cNum]
- ☐ Took[sID] ⊆ Student[sID]
- ☐ Took[oID] ⊆ Offering[oID]

Queries

1. Student ID of all students who have taken CS 5200.
   - $\pi_{sID}\ \sigma_{cNum=\text{"5200"} \wedge dept = \text{'CS'}}(Took \bowtie Offering)$
2. Student ID of all students who have taken CS 5200 and earned an 90 or higher in it.
   - $GoodStudents \leftarrow \Pi_{sID}\ \sigma_{dept=\text{"CS"} \wedge cNum=5200 \wedge grade \geq 90}(Took \bowtie Offering)$
3. The names of all such students
   - $\Pi_{surName,firstName}(GoodStudents \bowtie Student)$

Queries

1. Student ID of all students who have taken CS 5200.
   - $\pi_{sID}\ \sigma_{cNum=\text{"5200"} \wedge dept = \text{'CS'}}(Took \bowtie Offering)$
2. Student ID of all students who have taken CS 5200 and earned an 90 or higher in it.
   - $GoodStudents \leftarrow \Pi_{sID}\ \sigma_{dept=\text{"CS"} \wedge cNum=5200 \wedge grade \geq 90}(Took \bowtie Offering)$
3. The names of all such students
   - $\Pi_{surName,firstName}(GoodStudents \bowtie Student)$

More queries 4-7

4. The names of all students who have passed a breadth course with Professor Aoun.

5. sID of all students who have earned some grade over 80 and some grade below 50.

6. Terms when Miller and Mislove were both teaching something.

7. Terms when either of them was teaching CS 5200

More queries

4. The names of all students who have passed a breadth course with Professor Aoun.
   - AounBreadth ← $\Pi_{oID}\ \sigma_{breadth=true \wedge instructor=\text{“Aoun”}}$(Course ⋈ Offering)
   - Passers ← $\Pi_{sID}\ \sigma_{grade \geq 50}$(AounBreadth ⋈ T ook)
   - $\Pi_{surName,firstName}$(Passers ⋈ Student)
5. sID of all students who have earned some grade over 80 and some grade below 50.
   - $(\Pi_{sID}\sigma_{grade>80}\text{Took}) \cap (\Pi_{sID}\sigma_{grade<50}\text{Took})$
   - $(\Pi_{sID}\sigma_{grade>80}\text{Took}) \bowtie (\Pi_{sID}\sigma_{grade<50}\text{Took})$
   - why not $(\Pi_{sID}\sigma_{grade>80 \text{ and } grade<50}\text{Took})$ empty
   - why not $(\Pi_{sID}\sigma_{grade>80 \text{ or } grade<50}\text{Took})$ superset of answer
6. Terms when Miller and Mislove were both teaching something.

NO $\Pi_{term}\sigma_{instructor=\text{“Miller”} \text{ or } instructor = \text{“Mislove”}}\text{Offering}$

$\Pi_{term}\sigma_{instructor=\text{“Miller”}}\text{Offering} \cap \Pi_{term}\sigma_{instructor = \text{“Mislove”}}\text{Offering}$

More queries

6. Terms when Miller and Mislove were both teaching something.

Q7: change and to or -- now this works!

NO $\Pi_{term}\sigma_{instructor=\text{“Miller” and instructor = “Mislove”}}Offering$

$\Pi_{term}\sigma_{instructor=\text{“Miller”}}Offering \cap \Pi_{term}\sigma_{instructor = \text{“Mislove”}}Offering$

Q7: change intersection to union

7. Terms when either of them was teaching CS 5200
   - add in CS 5200 select

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

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.
  - $R \bowtie_{\theta} S$ where θ (greek theta) is any condition

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

  ×, ⋈

  ∩

  ∪, –

The highlighted operators are new. We'll learn them shortly.

- Unless very sure, use parentheses!

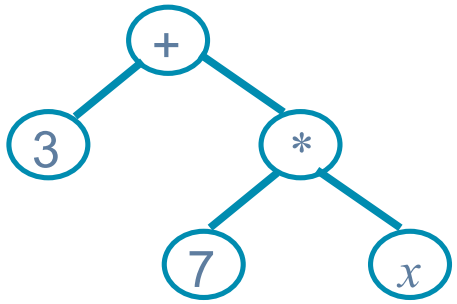
Breaking down expressions

- 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



Summary of 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	← (or :=)

More practice writing queries (9-12)

More queries

9. Terms when CS 5200 was not offered. (10 is similar)

$$(\Pi_{term}\text{Offering}) - (\Pi_{term}\sigma_{dept=\text{“CS”}\wedge cNum=5200}\text{Offering})$$

11. SIDs and surnames of all pairs of students who've taken a course together.

Self-Join!

$$T2(s2,o2,g2) \leftarrow \text{Took}$$
$$\text{Pairs}(sID1,sID2) \leftarrow \Pi_{sID,s2}\,(\sigma_{sID<s2\wedge oID=o2}(\text{Took} \times T2))$$
$$\text{OneName}(sID1,n1,sID2) \leftarrow \Pi_{sID1,surName,sID2}(\text{Pairs} \bowtie_{SID1=SID} \text{Student}))$$
$$\text{Answer}(sID1,n1,sID2,n2) \leftarrow \Pi_{sID1,name1,sID2,surName}\,(\text{OneName} \bowtie_{sID2=sID} \text{Student})$$

More queries

12. sID of student(s) with the highest grade in cs5200, in term 20099.

Can this be done in RA?

Let S be in answer.

Then, *for all* other students, their grades must be equal or lower

For all queries can be expressed in RA by converting into

not there exists

Find students who have a grade lower than some other student

(*there exists* a student with a higher grade)

Now negate this (we only want students not in this group)

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:
  $\{ t \mid t \in Movies \wedge t[director] = \text{“Scott”} \}$
- Expressive power (when limited to queries that generate finite results) is the same as RA. It is "relationally complete."