Relational Algebra

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

UNIVERSITY OF TORONTO

DCS50

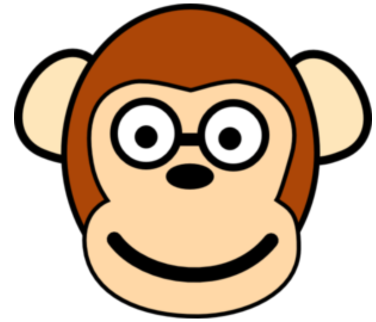
Announcements – Week 2

- If you're missing pre-requisites for CSC343 and we didn't discuss it yet ➔ email me (if you emailed, I'll get back to you)

- Assignment #1 will be posted next week (Relational Algebra queries)
  - Don't worry if you can't login to MarkUs yet, your accounts will be added soon
  - Plan your partnership!

- Today: introducing Relational Algebra + simple exercises
- Next week: more advanced RA examples ;-)

Questions from last week



- Is every <u>Key</u> also a <u>Superkey</u>, by definition?
  - Yes – it's just a special type of superkeys (minimal)
- Movies Schema: "Is there a limit to the # of directors a movie can have?"
  - Can't we just add a new tuple for the same movie, only with different director names?
  - Problem: from the database system's point of view, the only way of identifying a <u>unique</u> movie is using its <u>mID</u> field.

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

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

RA Basics

(covered by your week 2 Prep)

Elementary Algebra

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

UNIVERSITY OF TORONTO

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] ⊆ Movies[mID]
- Roles[aID] ⊆ 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?

Movies Schema

Movies(<u>mID</u>, title, director, year, length)
Artists(<u>aID</u>, aName, nationality)
Roles(<u>mID, aID, character</u>)

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

$\sigma_c(R)$

All British actors?
All movies from the 1970s?

Movies:

mID	title	director	year	length
1	Shining	Kubrick	1980	146
2	Player	Altman	1992	146
3	Chinatown	Polanski	1974	131
4	Repulsion	Polanski	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	nat
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

Exercise

- Write queries to find:
  - All British actors?

    $\sigma_{nat=\text{"British"}}(\text{Artists})$

  - All movies from the 1970s?

    $\sigma_{year>1969\ \wedge\ year<1980}\ (\text{Movies})$

$\sigma_{nat=\text{"British"}}(\text{Artists})$

aID	aName	nat
3	Stone	British

- What if we only want the names of all British actors? We need a way to pare down the columns.

Movies Schema

Movies(<u>mID</u>, title, director, year, length)
Artists(<u>aID</u>, aName, nationality)
Roles(<u>mID, aID, character</u>)

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

Movies:

mID	title	director	year	length
1	Shining	Kubrick	1980	146
2	Player	Altman	1992	146
3	Chinatown	Polanski	1974	131
4	Repulsion	Polanski	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	nat
1	Nicholson	American
2	Ford	American
3	Stone	British
4	Fisher	American

Roles:

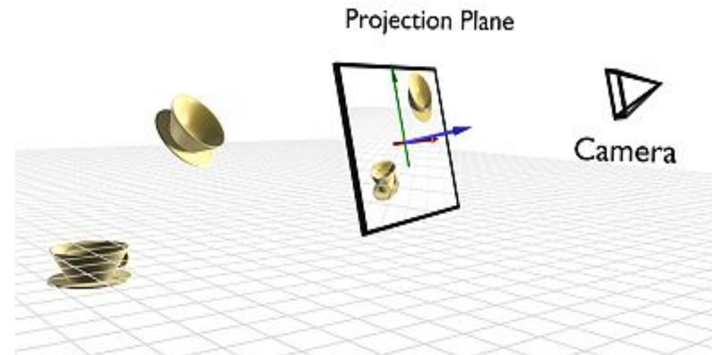
mID	**aID**	**character**
1	1	Jack Torrance
3	1	Jake 'J.J.' Gittes
1	3	Delbert Grady
5	2	Han Solo
6	2	Bob Falfa
5	4	Princess Leia Organa

Project: choose columns

- Notation: π_L (R)
  - R is a table.
  - L is a 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 π



- Why is it called "project"?
- What is the value of $\pi_{director}$ (Movies)?
  - (a) { Kubrick, Altman, Polanski, Polanski, Lucas, Lucas, Kubrick }, OR
  - (b) { Kubrick, Altman, Polanski, Lucas }

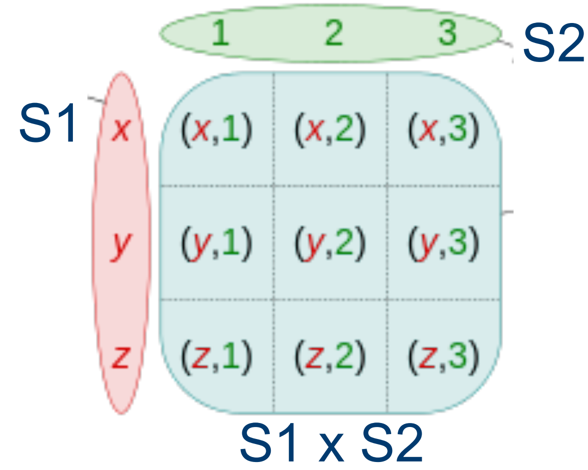
← *project* removes duplicates (think *sets*)

- Exercise: Write an RA expression to find the names of all directors of movies from the 1970s

$$\pi_{director}\ \sigma_{year>1969 \wedge year<1980}(Movies)$$

- 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 R1 followed by every attribute of R2, in order
- How many tuples are in R1 x R2?
  - *|R1| x |R2|*

|R|=cardinality of R

- Example: Movies x Roles
- If an attribute occurs in both relations, it occurs twice in the result (prefixed by relation name)
  - E.g.: **Movies**.mID, **Roles**.mID

Example of Cartesian product

profiles:

twitterID	name
hford	Harrison Ford
gclooney	George Clooney

follows:

aID	bID
hford	gclooney
gclooney	amal
marissamayer	lpage

profiles X follows:

twitterID	name	aID	bID
hford	Harrison Ford	hford	gclooney
hford	Harrison Ford	gclooney	amal
hford	Harrison Ford	marissamayer	lpage
gclooney	George Clooney	hford	gclooney
gclooney	George Clooney	gclooney	amal
gclooney	George Clooney	marissamayer	lpage

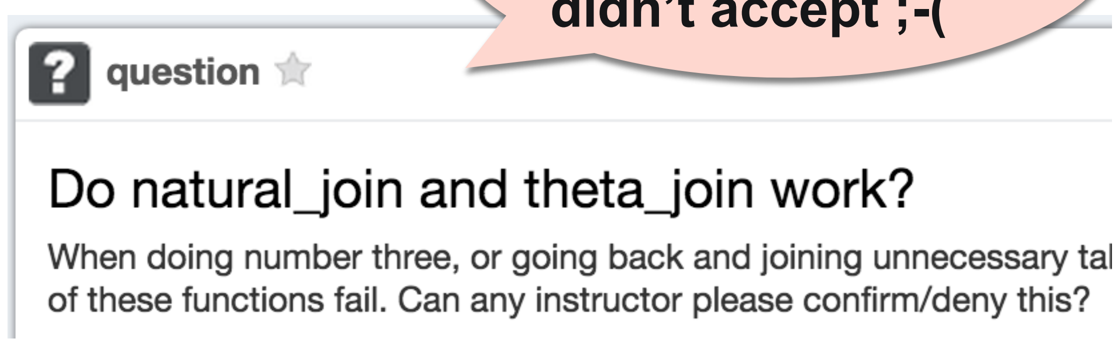
Cartesian product can be inconvenient

- It can introduce nonsense tuples.
- You can get rid of them with selects.

Just like you did in the lecture prep PCRS exercise ;-)

- But this is so highly common, an operation was defined to make it easier..
  - Joins..

Just like you tried to do but PCRS didn't accept ;-(



Joining two relations

Natural Join

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

Natural Join 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:

twitterID	name
hford	Harrison Ford
gclooney	George Clooney

follows:

aID	bID
hford	gclooney
gclooney	amal
marissamayer	lpage

profiles X follows:

twitterID	name	aID	bID
hford	Harrison Ford	hford	gclooney
hford	Harrison Ford	gclooney	amal
hford	Harrison Ford	marissamayer	lpage
gclooney	George Clooney	hford	gclooney
gclooney	George Clooney	gclooney	amal
gclooney	George Clooney	marissamayer	lpage

profiles:

twitterID	name
hford	Harrison Ford
gclooney	George Clooney

follows:

twitterID	bID
hford	gclooney
gcloon	
marissamay	lpage

profiles $\bowtie$ follows?

redundant attribute is omitted from result

twitterID	name	twitterID	bID
hford	Harrison Ford	hford	gclooney
~~hford~~	~~Harrison Ford~~	~~gclooney~~	~~amal~~
~~hford~~	~~Harrison Ford~~	~~marissamayer~~	~~lpage~~
~~gclooney~~	~~George Clooney~~	~~hford~~	~~gclooney~~
gclooney	George Clooney	gclooney	amal
~~gclooney~~	~~George Clooney~~	~~marissamayer~~	~~lpage~~

profiles:

twitterID	name
hford	Harrison Ford
gclooney	George Clooney

follows:

aID	twitterID
hford	gclooney
gclooney	amal
marissamayer	lpage

profiles ⋈ follows:

twitterID	name	aID	twitterID
~~hford~~	~~Harrison Ford~~	~~hford~~	~~gclooney~~
~~hford~~	~~Harrison Ford~~	~~gclooney~~	~~amal~~
~~hford~~	~~Harrison Ford~~	~~marissamayer~~	~~lpage~~
gclooney	George Clooney	hford	gclooney
~~gclooney~~	~~George Clooney~~	~~gclooney~~	~~amal~~
~~gclooney~~	~~George Clooney~~	~~marissamayer~~	~~lpage~~

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:

  $$R_1 \bowtie R_2 \bowtie \; ... \; \bowtie R_n$$

Questions

1. How many tuples are in Artists × Roles?
   – 24
2. How many tuples are in Artists ⋈ Roles?
   – 6
3. What is the result of:

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

4. What is the result of:

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

Movies Schema

Movies(<u>mID</u>, title, director, year, length)
Artists(<u>aID</u>, aName, nationality)
Roles(<u>mID, aID, character</u>)

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

Movies:

mID	title	director	year	length
1	Shining	Kubrick	1980	146
2	Player	Altman	1992	146
3	Chinatown	Polanski	1974	131
4	Repulsion	Polanski	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	nat
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

(Artists ⋈ Roles ⋈ Movies)

Artists. aID	Artists. aName	Artists. nat	Roles. MID	Roles. character	Movies. title	Movies. director	Movies. year	Movies. length
1	Nicholson	US	1	Jack Torrance	Shining	Kubrick	1980	146
1	Nicholson	US	3	Jack Gittes	Chinatown	Polanski	1974	131
2	Ford	US	5	Han Solo	Star Wars IV	Lucas	1977	126
2	Ford	US	6	Bob Falfa	American Graffiti	Lucas	1973	110
3	Stone	UK	1	Delbert Grady	Shining	Kubrick	1980	146
4	Fisher	US	5	Process Organa	Star Wars IV	Lucas	1977	126

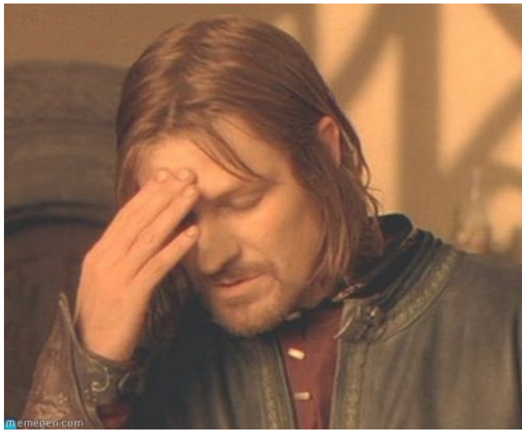
$\pi_{aName}\ \sigma_{director=\text{"Kubrick"}}$ (Artists ⋈ Roles ⋈ Movies) ?

aName
Nicholson
Stone

$(Artists \bowtie Roles \bowtie Movies)$

Artists. aID	Artists. aName	Artists. nat	Roles. MID	Roles. character	Movies. title	Movies. director	Movies. year	Movies. length
1	Nicholson	US	1	Jack Torrance	Shining	Kubrick	1980	146
1	Nicholson	US	3	Jack Gittes	Chinatown	Polanski	1974	131
2	Ford	US	5	Han Solo	Star Wars IV	Lucas	1977	126
2	Ford	US	6	Bob Falfa	American Graffiti	Lucas	1973	110
3	Stone	UK	1	Delbert Grady	Shining	Kubrick	1980	146
4	Fisher	US	5	Process Organa	Star Wars IV	Lucas	1977	126

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



Special cases for natural join

Imagine this scenario

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

Result of natural join

Artist	Name
5555	Patrick Stewart
1868	Angelina Jolie

Artist	Name
1234	Brad Pitt
1868	Angelina Jolie
5555	Patrick Stewart

Exactly the same attributes

What about this one?

Artist	Name
1234	Brad Pitt
1868	Angelina Jolie
5555	Patrick Stewart

No attributes in common

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

Result? **Cartesian product** of the two relations!

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?

  …use a Cartesian product + select ;-)

UNIVERSITY OF TORONTO

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! (which Artists are also Directors?)
- Solution?

Theta Join

- It's common to use σ to check conditions after a Cartesian product.

$$\boxed{\sigma_{condition}\ (R \times S)}$$

- 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, $R \bowtie_{condition} S = \sigma_{condition}\ (R \times S)$.

Theta Join

$R \bowtie_{condition} S$

- The word "theta" has no special connotation.
  It is an artifact of a definition in an early paper.
- You still have to write out the *conditions*, since they are not inferred.

- Exercise: Use Theta join to find artists who have also *directed* movies. Display artist name, movie title.

$\pi_{aName,title}(Artists \bowtie_{Artists.aName=Movies.director} Movies)$

Outer Joins

Dangling tuples

- If a tuple in one relation has no match in the other, natural join leaves that tuple out.
- Example schema:
  - People(<u>phone</u>, name, address)
  - Donations(<u>phone, charity, </u>amount, date)
    Phone is a foreign key referencing People.
- What if someone has made no donations?
- Natural join leaves out those tuples, but you may want them!

UNIVERSITY OF TORONTO

Outer Join

- Outer Join leaves those tuples in, and adds *null* values whenever no value exists.
- Three variants:
  - Left Outer Join: only do this "null padding" for left operand
  - Right Outer Join: only do this null padding for right operand
  - Full Outer Join: pad both operands! (see next slide for example)

Example of Left Outer Join

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

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

Full Outer Join - Example

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

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
ω	ω	Production	Charles

Composing larger expressions (plus a few new operators)

Assignment operator

- Notation:

  R := Expression

- Alternate notation:

  $R(A_1, \ldots, A_n)$:= 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.

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!

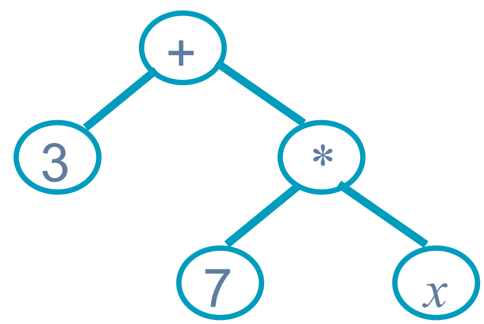
Breaking down expressions

- 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

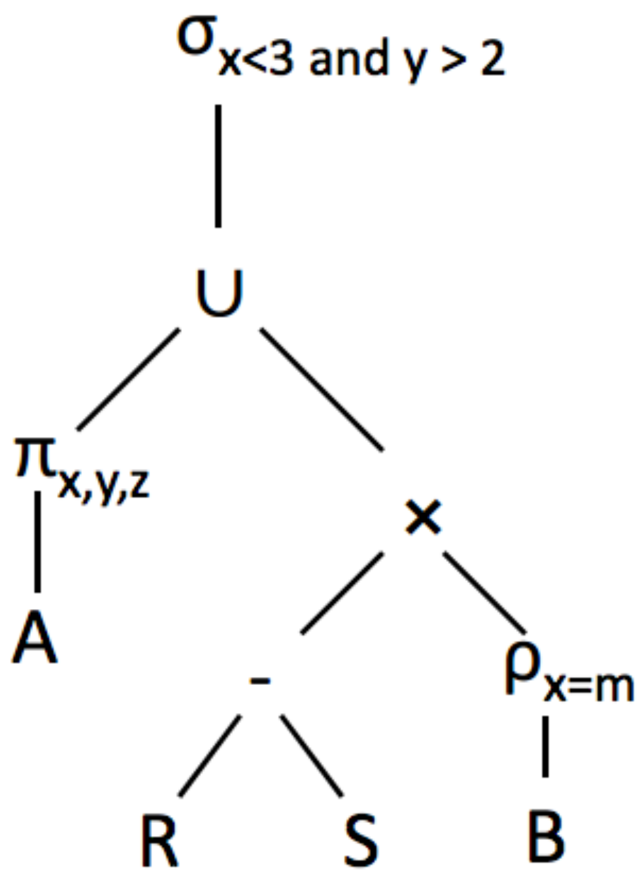
- Leaves are relations.
- Interior notes are operators.
- Exactly like representing arithmetic expressions as trees.

$3 + 7 * x$



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

Expression Trees – RA Example



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.

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).

Union

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates ∪ Managers

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38
9297	O'Malley	56

Intersection

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates ∩ Managers

Number	Surname	Age
7432	O'Malley	39
9824	Darkes	38

Example from: http://www.cs.toronto.edu/~faye/343/f07/lectures/wk3/03_RAlgebra.pdf

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).

Difference

Graduates

Number	Surname	Age
7274	Robinson	37
7432	O'Malley	39
9824	Darkes	38

Managers

Number	Surname	Age
9297	O'Malley	56
7432	O'Malley	39
9824	Darkes	38

Graduates - Managers

Number	Surname	Age
7274	Robinson	37

Example from: http://www.cs.toronto.edu/~faye/343/f07/lectures/wk3/03_RAlgebra.pdf

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*?

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Maternity

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

Paternity ∪ Maternity ???

Example from: http://www.cs.toronto.edu/~faye/343/f07/lectures/wk3/03_RAlgebra.pdf

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*?

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Maternity

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

$\rho_{Father\text{->}Parent}(\textbf{Paternity}) \cup \rho_{Mother\text{->}Parent}(\textbf{Maternity})$

Parent	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

Example from: http://www.cs.toronto.edu/~faye/343/f07/lectures/wk3/03_RAlgebra.pdf

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):
  - 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.

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 more efficient.
- But in RA, we won't care about efficiency;
  it's an algebra.
- In a DBMS, queries actually are executed, & 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.