CMPT 321
Fall 2017

Relational algebra queries

Lecture 02.02

By *Marina Barsky*

Relational algebra queries

- Any query can be expressed using core operators: σ, π, x, U, –, ρ
- Derived operators: ∩, ⋈
- Any RA Operator returns relation, so we can compose complex queries from known operators

RA has Limitations !

- Cannot compute “transitive closure”

Name1	Name2	Relationship
Fred	Mary	Father
Mary	Joe	Cousin
Mary	Bill	Spouse
Nancy	Lou	Sister

- Find all direct and indirect relatives of Fred
- Cannot express in RA !!!
  - Need to write C program, use a graph engine, or PL-SQL…

APPROACHES TO WRITING RELATIONAL ALGEBRA QUERIES

Running example: Movies database

Movie (title, year, length, inColor, studioName, producerC)

MovieStar (name, address, gender, birthdate)

StarsIn (movieTitle, movieYear, starName)

MovieExec (name, address, cert, netWorth)

Studio (studioname, presc);

Rules for simple queries

1. Find producer of ‘Star wars’
2. Title and length of all Disney movies produced in year 1990
3. For each movie’s title produce the name of this movie’s producer

Rule 1

- *Ask yourself which relations need to be involved. Ignore the rest!*
- *Every time you combine relations, confirm that you specify the names of matching attributes (unless natural join)*

Rule 2. Write intermediate relations with attributes and sample data

- *Remember that selection checks one tuple at a time.*
- *If you need info from two different tuples, you must make a new relation where all the required info is in one tuple.*
- *Use variable assignment to define this intermediate relation.*
- *To visualize:*
  - *Draw an example of an intermediate relation with actual data in it.*
  - *Use good names for 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.*

Movie (title, year, length, inColor, studioName, producerC)

MovieStar (name, address, gender, birthdate)

StarsIn (movieTitle, movieYear, starName)

MovieExec (name, address, cert, netWorth)

Studio (studioname, presc);

4. Find all name pairs in form (movie star, movie producer) that live at the same address.

$$Star = \rho_{star,staraddress}(\pi_{name,\ address}(MovieStar))$$
$$Prod = \rho_{prod,\ prodaddress}(\pi_{name,\ address}(MovieExec))$$

$$\pi_{star,prod}((Star) \bowtie_{staraddress=prodaddress\ AND\ star\ !=prod} (Prod))$$

Movie (title, year, length, inColor, studioName, producerC)
MovieStar (name, address, gender, birthdate)
StarsIn (movieTitle, movieYear, starName)
MovieExec (name, address, cert, netWorth)
Studio (studioname, presc)

5. Find all movies where there were only male actors.

MaleMovies_InclFemales(title, year) =
$\pi_{title,year}(\sigma_{gender='male'}(MovieStar) \bowtie StarsIn))$

MM_F
title
~~A~~
B
C
~~D~~

-

FM_M
title
A
D
E
F

FemaleMovies_InclMales(title, year) =
$\pi_{title,year}(\sigma_{gender='female'}(MovieStar) \bowtie StarsIn))$

MoviesMalesOnly = MM_F – FM_M

MM
title
B
C

Movie (title, year, length, inColor, studioName, producerC)

MovieStar (name, address, gender, birthdate)

StarsIn (movieTitle, movieYear, starName)

MovieExec (name, address, cert, netWorth)

Studio (studioname, presc);

6. Find the names of all producers who did NOT produce 'Star wars'

- Simple: $\pi_{name}(\text{MovieExec}) - \pi_{name}((\text{Movie}) \bowtie_{title='Star\ wars'\ AND\ producerC=cert} (\text{MovieExec}))$

- More efficient (smaller Cartesian product)

$$\pi_{name}((\sigma_{title='Star\ wars'}(\text{Movie})) \bowtie_{producerC!=cert} (\text{MovieExec}))$$

Movie (title, year, length, inColor, studioName, producerC)

MovieStar (name, address, gender, birthdate)

StarsIn (movieTitle, movieYear, starName)

MovieExec (name, address, cert, netWorth)

Studio (studioname, presc);

7. Find names of producers that produced at least one movie for each of different studios: Disney and MGM

$$\pi_{name}[(\sigma_{studioName='Disney'}(Movie)) \bowtie_{producerC=cert} (MovieExec)]$$

$$\wedge$$

$$\pi_{name}[(\sigma_{studioName='MGM'}(Movie)) \bowtie_{producerC=cert} (MovieExec)]$$

Movie (title, year, length, inColor, studioName, producerC)

MovieStar (name, address, gender, birthdate)

StarsIn (movieTitle, movieYear, starName)

MovieExec (name, address, cert, netWorth)

Studio (studioname, presc);

8. Find all movie titles for which there is no producer entry in MovieExec table

$\pi_{title}(Movie) - \pi_{title}((Movie) \bowtie_{producerC=cert} (MovieExec))$

Rule 3. Computing Max (min is analogous)

- *Do self-product and find those that are not max*
- *Subtract from all to find the maxes*

Movie (title, year, length, inColor, studioName, producerC)
MovieStar (name, address, gender, birthdate)
StarsIn (movieTitle, movieYear, starName)
MovieExec (name, address, cert, netWorth)
Studio (studioname, presc)

9. Find movie producers with max net worth. In case of ties, return the list of all such top producers.

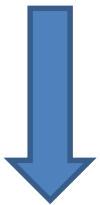
- Rename MovieExec

Producers1 (name, networth) = ρ $[\pi_{name,networth}(MovieExec)]$

Producers2 (name, networth) = ρ $[\pi_{name,networth}(MovieExec)]$

- Pair all tuples in P1 with all other tuples in P2 (Cartesian product, not a join):

P1 x P2



- Select those that are not max, because there are some pairs where P1.networth < P2.networth

$NotTop = \sigma_{P1.networth<P2.networth}(P1 \times P2)$

$Result = \pi_{name}MovieExec - \pi_{name}(NotTop)$

Rule 4. Queries asking for “every”

- *Make all combinations that include both every and some*
- *Subtract those that make it “not every”. The result is those who failed “every”.*
- *Subtract the failures from all to get a result*

Movie (title, year, length, inColor, studioName, producerC)
MovieStar (name, address, gender, birthdate)
StarsIn (movieTitle, movieYear, starName)
MovieExec (name, address, cert, netWorth)
Studio (studioname, presc)

10. Find the names of movie stars who starred in **every** Disney movie

- First, for each star – all Disney movies they starred in:

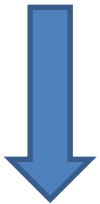
StarsInDisney = $\pi_{name,title}$ (StarsIn⋈Movie$_{studioName='Disney'}$)

SD	
name	title
A	D
A	E
B	D
C	E

- Now need to find those stars who are **not** in **every** DisneyMovie, missing some.

- For this, we create all combinations of the above actors with the above movies:

allCombSD = π_{name} SD x π_{title} SD



allComb	
name	title
A	D
A	E
B	D
B	E
C	D
C	E

allComb	
name	title
A	D
A	E
B	D
B	E
C	D
C	E

-

SD	
name	title
A	D
A	E
B	D
C	E

- How to remove those who are not in every movie?

NotEvery = π_{name}(allComb−SD)

- Finally obtain stars who starred in every Disney movie:

everyDisney = π_{name}(SD) - NotEvery

Rule 5. K or more

- *Make k Cartesian products with itself and select rows where all k values are equal*

Movie (title, year, length, inColor, studioName, producerC)

MovieStar (name, address, gender, birthdate)

StarsIn (movieTitle, movieYear, starName)

MovieExec (name, address, cert, netWorth)

Studio (studioname, presc);

11. Find the names of all stars which starred in at least 2 movies (according to our database)

1. $S1=\rho_{title1,year1,name1}(StarsIn)$

   $S2=\rho_{title2,year2,name2}(StarsIn)$

2. $(S1) \bowtie_{name1=name2 \text{ AND } (title1 \text{ != } title2 \text{ or } year1!=year2)} (S2)$

Rule 6. Exactly k

- “k or more” – “(k+1) or more”

Movies

EVEN MORE COMPLEX QUERIES

12. Find all name pairs in form (movie star, movie producer) that live at the same address. The same person can be both a star and a producer. Now, try to eliminate palindrome pairs: leave (a,b) but not both (a,b) and (b,a).

12 – solution 1. Find all name pairs in form (movie star, movie producer) that live at the same address. The same person can be both a star and the producer. Now, try to eliminate palindrome pairs: leave (a,b) but not both (a,b) and (b,a).

1. $Star = \rho_{name \rightarrow star}(MovieStar)$

   $Prod = \rho_{name \rightarrow prod}(MovieExec)$

2. $Pairs = \pi_{star,prod}((Star) \bowtie_{Star.address=Prod.address\ AND\ star!=prod} (Prod))$

3. $PA = \sigma_{star<prod}(Pairs)$ // Pairs in Ascending order

   $PD = \sigma_{star>prod}(Pairs)$ //Pairs in Descending order

4. $Palindrome = (PA) \bowtie_{PA.star=PD.prod\ AND\ PA.prod=PD.star} (PD)$
5. $Pairs - \pi_{PD.star,PD.prod}(Palindrome)$

Example on the next page

Step 1. Renaming

Star	
star	addr
A	1
B	1
C	2
F	3

Prod	
prod	addr
A	1
B	1
D	2
E	3

1

Star=$\rho_{name \rightarrow star}$(MovieStar)

Prod=$\rho_{name \rightarrow prod}$(MovieExec)

Star	Addr	Prod	Addr
A	1	A	1
A	1	B	1
A	1	D	2
A	1	E	3
B	1	A	1
B	1	B	1
B	1	D	2
B	1	E	3
C	2	A	1
C	2	B	1
C	2	D	2
C	2	E	3
F	3	A	1
F	3	B	1
F	3	D	2
F	3	E	3

Step 2. Cartesian product: Star x Prod

2. Pairs = $\pi_{star,prod}$ ((Star) $\bowtie_{Star.address=Prod.address\ AND\ star!=prod}$ (Prod))

Pairs	
Star	Prod
A	B
B	A
C	D
F	E

Step 3. Sorted pairs

Pairs	
Star	Prod
A	B
B	A
C	D
F	E

3. PA = $\sigma_{star<prod}$(Pairs) // Pairs where Star < Prod
   PD = $\sigma_{star>prod}$(Pairs) //Pairs where Star > Prod

PA	
Star	Prod
A	B
C	D

PD	
Star	Prod
B	A
F	E

Step 4. Cartesian product PA x PD

PA	
Star	Prod
A	B
C	D

x

PD	
Star	Prod
B	A
F	E

Palyndrome (only colored tuple qualify)			
PA.Star	PA.Prod	PD.Star	PD.Prod
A	B	B	A
A	B	F	E
C	D	B	A
C	D	F	E

4. Palindrome = (PA) $\bowtie_{\text{PA.star=PD.prod AND PA.prod=PD.star}}$ (PD)

Step 5. Remove palindrome tuples from pairs

5. Pairs – $\pi_{PD.star,PD.prod}$ (Palindrome)

Pairs	
Star	Prod
A	B
B	A
C	D
F	E

-

Palyndrome			
PA.Star	PA.Prod	PD.Star	PD.Prod
A	B	B	A

result	
Star	Prod
A	B
C	D
F	E

12. Another solution proposed in class

$Star = \rho_{name \rightarrow star}(MovieStar)$
$Prod = \rho_{name \rightarrow prod}(MovieExec)$

$SP = Star \bowtie_{Star.address=Prod.address \text{ AND } star!=prod} Prod$
$PS = Prod \bowtie_{Star.address=Prod.address \text{ AND } star!=prod} Star$

$PAIRS = \rho_{star \rightarrow name1,\ prod \rightarrow name2}(SP) \cup \rho_{prod \rightarrow name1,\ star \rightarrow name2}(PS)$

$Result = \sigma_{name1 < name2}(Pairs)$

Example on the next page

Step 1. Renaming

The renaming is done for readability - to distinguish names:

MovieStar.name → Star.star

MovieExec.name → Prod.prod

Star	
star	addr
A	1
B	1
C	2
F	3

Prod	
prod	addr
A	1
B	1
D	2
E	3

$Star = \rho_{name \rightarrow star}(MovieStar)$

$Prod = \rho_{name \rightarrow prod}(MovieExec)$

Step 2. Join Star ⋈ Prod and Prod ⋈ Star on address

$SP = Star \bowtie_{Star.address=Prod.address \text{ AND } star!=prod} Prod$

$PS = Prod \bowtie_{Star.address=Prod.address \text{ AND } star!=prod} Star$

Star	
star	addr
A	1
B	1
C	2
F	3

Prod	
prod	addr
A	1
B	1
D	2
E	3

SP	
star	prod
A	B
B	A
C	D
F	E

PS	
prod	star
A	B
B	A
D	C
E	F

Step 3. Union (set union) of SP and PS

PAIRS = $\rho_{\text{star} \rightarrow \text{name1}, \text{prod} \rightarrow \text{name2}}(\text{SP}) \cup \rho_{\text{prod} \rightarrow \text{name1}, \text{star} \rightarrow \text{name2}}(\text{PS})$

SP	
star	prod
A	B
B	A
C	D
F	E

PS	
prod	star
A	B
B	A
D	C
E	F

PAIRS	
name1	name 2
A	B
B	A
C	D
F	E
D	C
E	F

Step 4. Select only one instance of palindrome pair – where name1<name2

Result = $\sigma_{name1 < name2}$ (Pairs)

PAIRS	
name1	name 2
A	B
B	A
C	D
F	E
D	C
E	F

Result	
name1	name 2
A	B
C	D
E	F

We don't know at this point who is a star and who is a producer, but we can later do the selection for each name in both tables to figure out if this is important for our query

Homework database: Pizza

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

Pizza

TEST YOURSELF ON SIMPLE QUERIES

Projections: Pizza

1. Find full information about all possible places and prices to get mushroom or pepperoni pizzas

2. Find name of pizzerias that serve mushroom or pepperoni pizzas

3. Compute the full list of pizza types, with the corresponding pizzerias and the price of pizza in cents

Selections: Pizza

4. Find names of all customers under 18
5. Find names of all female customers older than 25

Join: Pizza

6. Find all pizza types that both Amy and Dan eat

7. Find the names of all females who eat a mushroom pizza

8. Find the names of pizzerias where Hil can buy pizzas she eats for less than 10$