Relational-algebra exercises

Appendix to Lecture 3

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);

Movies

SIMPLE QUERIES

Selections: Movies

1. Find titles of all black-and-white movies which were produced after 1970

2. Find titles of all movies produced by MGM studio after 1970 or with length less than 1.5 hours

3. Find producer of ‘Star wars’

Projections: Movies

4. Info about all Disney movies produced in year 1990

5. Title and length of all Disney movies produced in year 1990

6. Title and length in hours of all Disney movies produced in year 1990

Joins: Movies

7. For each movie’s title produce the name of this movie’s producer

8. Find the names of producers of movies where Harrison Ford starred.

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

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

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

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

Movies

MORE COMPLEX QUERIES

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

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

- Simple:

$\pi_{name}(MovieExec) -$

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

- More efficient (smaller Cartesian product)

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

**9B. Find all name pairs in form (movie star, movie producer) that live at the same address. 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

Example on the next page

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

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

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

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

2. Cartesian product: Star x Prod

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

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

3. Sorted pairs

Pairs	
Star	Prod
A	B
B	A
C	D
F	E

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

PA	
Star	Prod
A	B
C	D

PD	
Star	Prod
B	A
F	E

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)

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

Movie (<u>title</u>, <u>year</u>, length, inColor, studioName, producerC)

MovieStar (<u>name</u>, address, gender, birthdate)

StarsIn (<u>movieTitle</u>, <u>movieYear</u>, <u>starName</u>)

MovieExec (<u>name</u>, address, cert, netWorth)

Studio (<u>studioname</u>, presc);

11. 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);

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

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

Movie (<u>title</u>, <u>year</u>, length, inColor, studioName, producerC)

MovieStar (<u>name</u>, address, gender, birthdate)

StarsIn (<u>movieTitle</u>, <u>movieYear</u>, <u>starName</u>)

MovieExec (<u>name</u>, address, cert, netWorth)

Studio (<u>studioname</u>, presc);

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

Lab 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$

Person (name, age, gender)
Frequents (name, pizzeria)
Eats (name, pizza)
Serves (pizzeria, pizza, price)

9. Find the names of all females who eat either mushroom or pepperoni pizza (or both).

$$\pi_{name}(\sigma_{gender='female' \text{ AND } (pizza='mushroom' \text{ OR } pizza='pepperoni')}(Person \bowtie Eats))$$

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

10. Find the names of all females who eat both mushroom and pepperoni pizza.

$$\pi_{name}(\sigma_{gender='female' \text{ AND } pizza='mushroom'}(Person \bowtie Eats))$$

$$\cap$$

$$\pi_{name}(\sigma_{gender='female' \text{ AND } pizza='pepperoni'}(Person \bowtie Eats))$$

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

11. Find all pizzerias that serve at least one pizza that Amy eats for less than $10.00.

$\pi_{pizzeria}(\sigma_{name='Amy'}(Eats) \bowtie \sigma_{price<10}(Serves))$

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

12. Find all pizzerias frequented by at least one person under the age of 18.

$$\pi_{pizzeria}(\sigma_{age<18}(Person) \bowtie Frequents)$$

Person (<u>name</u>, age, gender)

Frequents (<u>name</u>, <u>pizzeria</u>)

Eats (<u>name</u>, <u>pizza</u>)

Serves (<u>pizzeria</u>, <u>pizza</u>, price)

13. Find all pizza types which are not eaten by anyone

$\pi_{pizza}(Serves) - \pi_{pizza}(Eats)$

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

14. Find all pizzerias that are frequented by only females or only males.

$$\left[\pi_{pizzeria}(\sigma_{gender='female'}(Person) \bowtie Frequents) - \pi_{pizzeria}(\sigma_{gender='male'}(Person) \bowtie Frequents)\right]$$

$$\cup$$

$$\left[\pi_{pizzeria}(\sigma_{gender='male'}(Person) \bowtie Frequents) - \pi_{pizzeria}(\sigma_{gender='female'}(Person) \bowtie Frequents)\right]$$

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

15. Find all pizzerias where Dan could buy pizzas that he eats, and where he has never bought a pizza yet

$\pi_{pizzeria}[(\sigma_{name='Dan'}(Eats)) \bowtie (Serves)]$

-

$\pi_{pizzeria}(\sigma_{name='Dan'}(Frequents))$

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

16. For each person, find all pizzas the person eats that are not served by any pizzeria the person frequents. Return all such person (name) / pizza pairs.

$Eats - \pi_{name,pizza}(Frequents \bowtie Serves)$

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

17. Find the names of all people who frequent only pizzerias serving at least one pizza they eat.

$$\pi_{name}(\text{Person})$$

$$-$$

$$\pi_{name}(\text{Frequents} - \pi_{name,pizzeria}(\text{Eats} \bowtie \text{Serves}))$$

Explanation on the next page

17. Find the names of all people who frequent only pizzerias serving at least one pizza they eat.

- 1. List of all pizzerias which serve at least one of pizzas which particular person can eat:

$\pi_{name,pizzeria}(Eats \bowtie Serves)$

- 2. List of all pizzerias which are frequented by this person but do not serve any pizza he can it

$Frequents - \pi_{name,pizzeria}(Eats \bowtie Serves)$

- 3. Answer to the query

$\pi_{name}(Person)$

$-$

$\pi_{name}(Frequents - \pi_{name,pizzeria}(Eats \bowtie Serves))$

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

18. Find the names of all people who frequent every pizzeria serving at least one pizza they eat.

Explanation on the next page

$$\pi_{name}(\text{Person})$$

$$-$$

$$\pi_{name}(\pi_{name,pizzeria}(\text{Eats} \bowtie \text{Serves}) - \text{Frequents})$$

18. Find the names of all people who frequent every pizzeria serving at least one pizza they eat.

- 1. List of all pizzerias per person which serve at least one pizza this person can eat:

$\pi_{name,pizzeria}(Eats \bowtie Serves)$

- 2. List of pizzerias which serve the desirable pizza but which person did not visit yet

$\pi_{name,pizzeria}(Eats \bowtie Serves) - Frequents$

- 3. All the people excluding those in p.2

$\pi_{name}(Person)$

$-$

$\pi_{name}(\pi_{name,pizzeria}(Eats \bowtie Serves) - Frequents)$

Person (name, age, gender)

Frequents (name, pizzeria)

Eats (name, pizza)

Serves (pizzeria, pizza, price)

19. Find the pizzeria serving the cheapest pepperoni pizza. In the case of ties, return all of the cheapest-pepperoni pizzerias.

$\pi_{pizzeria}(\sigma_{pizza='pepperoni'}Serves)$

$-$

$\pi_{pizzeria}\ [\sigma_{price>price2}($

$\pi_{pizzeria,price}(\sigma_{pizza='pepperoni'}Serves)$

$\times$

$\rho_{pizzeria2,price2}\pi_{pizzeria,price}(\sigma_{pizza='pepperoni'}Serves))]$

Explanation on the next page

19. Find the pizzeria serving the cheapest pepperoni pizza. In the case of ties, return all of the cheapest-pepperoni pizzerias.

- 1. Finds all pizzerias where price for pepperoni pizza is greater than in some other pizzeria

$$\sigma_{price>price2}\Big(\pi_{pizzeria,price}(\sigma_{pizza='pepperoni'}Serves) \times \rho_{pizzeria2,price2}[\pi_{pizzeria,price}(\sigma_{pizza='pepperoni'}Serves)] \Big)$$

- 2. Subtracts it from all other pizzerias serving pepperoni pizzas

$$\pi_{pizzeria}(\sigma_{pizza='pepperoni'}Serves) - \pi_{pizzeria}\ [\sigma_{price>price2}(\pi_{pizzeria,price}(\sigma_{pizza='pepperoni'}Serves) \times \rho_{pizzeria2,price2}\pi_{pizzeria,price}(\sigma_{pizza='pepperoni'}Serves))]$$