Writing relational algebra queries

More general approach

1. Choose relations involved

- 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)

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

Example 1

- Find all pizzerias frequented by at least one person under the age of 18.
- We need pizzeria and age, joined on name
- Creating a new relation:

ExtendedFrequents (name, age, gender, pizzeria) = Person $\bowtie$ Frequents

- We then apply the selection to ExtendedFrequents:

$\sigma_{age<18}$ ExtendedFrequents

- And finally projection for desirable attributes

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

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 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.

Example 2

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

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

PizzeriasFemales_MalesNotExcluded(pizzeria)
=
$\pi_{pizzeria}(\sigma_{gender='female'}(Person) \bowtie Frequents))$

PizzeriaMales_FemalesNotExcluded(pizzeria)
=
$\pi_{pizzeria}(\sigma_{gender='male'}(Person) \bowtie Frequents))$

PizzeriaFemalesOnly = PF_M – PM_F

PF_M
pizzeria
~~A~~
B
C
~~D~~

-

PM_F
pizzeria
A
D
E
F

PFemales
pizzeria
B
C

3. Computing Max (min is analogous)

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

Example 3: 1/2

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

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

ServesPepperoni1 (pizzeria, price) = $\rho_{ServesPepperoni1}$

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

ServesPepperoni2 (pizzeria, price) = $\rho_{ServesPepperoni2}$

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

- Pair all tuples in SP1 with all other tuples in SP2 (Cartesian product, not a join):

ServesPepperoni1 x ServesPepperoni2

Example 3: 2/2

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

- Select those that are not min price, because there are some pairs where SP1.price > SP2.price

$\text{PizzeriasNotChepestPP} = \sigma_{SP1.price > SP2.price2}(SP1 \times SP2)$

$\text{Result} = \pi_{pizzeria}\,\text{ServesPepperoni1} - \pi_{SP1.pizzeria}(\,\text{PizzeriasNotChepestPP})$

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

Example 4

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

- Find the names of all people who frequent **every** pizzeria serving at least one pizza they eat.
- First, for each person – all pizzerias that serve pizzas the person eats

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

- Now need to find those people who do **not** frequent **every** good pizzeria, missing some that serve desirable pizzas:

NotEvery = $\pi_{name}(PotentialGoodPizzerias - Frequents)$

Frequent**Every**GoodPizzeria = $\pi_{name}(Person)$ - NotEvery

5. K or more

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

Example 5

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

- Find names of all pizzerias which serve **at least 2** pizzas that Amy can eat
- Pizzerias which serve desirable pizzas (**including** those that serve **only 1 good** pizza)

AllAmyPizzerias = $\pi_{\text{pizzeria, pizza}}\,(\sigma_{\text{name='Amy'}}(\text{Eats}) \bowtie \text{Serves})$

- Cartesian product of AAP with itself, select only rows where pizzerias are equal – and pizzas are different

A1 = ρ_{A1}(AllAmyPizzerias), A2 = ρ_{A2}(AllAmyPizzerias)

AtLeast2 = $\sigma_{\text{A1.pizza>A2.pizza AND A1.pizzeria = A2.pizzeria}}(\text{A1 x A2})$

Answer = $\pi_{\text{pizzeria}}(\text{AtLeast2})$

6. Exactly k

- “k or more” – “(k+1) or more”

Example 6

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

- Find the names of all pizzerias which serve **exactly 2** pizzas that Amy can eat

A1 = ρ_{A1}(AllAmyPizzerias), A2 = ρ_{A2}(AllAmyPizzerias), A3 = ρ_{A3}(AllAmyPizzerias)

- We have AtLeast2
- Compute at least 3:

$$\text{AtLeast3} = \sigma_{\substack{\text{A1.pizza>A2.pizza AND A1.pizzeria = A2.pizzeria} \\ \text{AND A1.pizza>A3.pizza AND A1.pizzeria = A3.pizzeria}}} (\text{A1 x A2 x A3})$$

Exactly2 = AtLeast2 – AtLeast3

Answer = $\pi_{pizzeria}$ (Exactly2)

7. Related pairs

- Do Cartesian product and select the non-equal pairs joined on the desired shared attribute

Example 7

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

- Find all pairs of customers who frequent the same pizzeria

$F1 = \rho_{F1}(\text{Frequents})$

$F2 = \rho_{F2}(\text{Frequents})$

$\sigma_{F1.pizzeria=F2.pizzeria \text{ AND } F1.name < F2.name}(F1 \times F2)$