Tutorial: Relational Algebra

This tutorial uses the schema from Pizza. You are encouraged to use the relational algebra evaluator[1] for testing. Alternatively, you may simply write a more relaxed version of relational algebra that is closer to SQL query.

1. Translate the following queries into relational algebra.
   (a) Find all pizzas that Alice likes but not pizzas that Bob likes.

Solution: We can first think of the SQL query first or we can solve this directly using algebra by decomposition. If we want to use decomposition, we can decompose the problem into two problems:

- Find all pizzas that Alice likes: $\pi_{[\text{pizza}]}(\sigma_{[\text{cname}=\text{'Alice'}]}(\text{likes}))$
- Find all pizzas that Bob likes: $\pi_{[\text{pizza}]}(\sigma_{[\text{cname}=\text{'Bob'}]}(\text{likes}))$

Then we compose the solution back by *subtracting* the pizzas that Bob likes. This gives us the following solution.

$$\pi_{[\text{pizza}]}(\sigma_{[\text{cname}=\text{'Alice'}]}(\text{likes})) - \pi_{[\text{pizza}]}(\sigma_{[\text{cname}=\text{'Bob'}]}(\text{likes}))$$

If we want, we can also find an SQL solution first.

```
(SELECT l.pizza
 FROM likes l
 WHERE l.cname = 'Alice')
EXCEPT
(SELECT l.pizza
 FROM likes l
 WHERE l.cname = 'Bob');
```

The SQL above maps directly to the relational algebra. Unfortunately, if your SQL solution has nested query, it cannot be easily converted to relational algebra.

```
SELECT l.pizza
FROM likes l
WHERE l.cname = 'Alice'
  AND l.pizza NOT IN (
    SELECT l.pizza
    FROM likes l
    WHERE l.cname = 'Bob'
  );
```

[1] https://relational-algebra-evaluator.onrender.com/

(b) Find all customer-restaurant pairs (C, R) where C and R both located in the same area and C likes some pizza that is sold by R where the price of the pizza is less than \$15.

Solution: A naïve solution is to use the following simple query.

```
SELECT DISTINCT c.cname, r.rname
FROM customer c, likes l, sells s, restaurant r
WHERE c.cname = l.cname AND s.rname = r.rname
  AND l.pizza = s.pizza AND c.area = r.area
  AND s.price < 15;
```

Unfortunately, a direct translation of this into relational algebra will take a long time on the evaluator. But if we use natural join (i.e., $\bowtie$), then the computation can finish much quicker.

$$\pi_{[\text{cname, rname}]}(\sigma_{[(\text{price} < 15)]}(\text{customer} \bowtie \text{likes} \bowtie \text{sells} \bowtie \text{restaurant}))$$

There is another issue with direct translation to be run on the evaluator and that is the renaming has to be done on the column name. As such, we will not show the full code using renaming. Instead, we will show the alternative solution if we can rename the table and use the dot notation.

$$\begin{aligned}&\pi_{[\text{c.cname, r.rname}]}(\\ &\quad\sigma_{[(\text{c.cname} = \text{l.cname}) \wedge (\text{s.rname} = \text{r.rname}) \wedge (\text{l.pizza} = \text{s.pizza}) \wedge (\text{c.area} = \text{r.area}) \wedge (\text{s.price} < 15)]}(\\ &\qquad\rho(\text{customer}, \text{c}) \times \rho(\text{likes}, \text{l}) \times \rho(\text{sells}, \text{s}) \times \rho(\text{restaurant}, \text{r})\\ &\quad))\end{aligned}$$

Notice the mapping to the SQL query.

There is another solution that involve *intersection*. This can be achieved by decomposing the problem into two parts:

- Find all customer and restaurant pairs in the same area: $\pi_{[\text{cname, rname}]}(\text{customer} \bowtie \text{restaurant})$
- Find all customer and restaurant pairs such that the restaurant sells some pizza that the customer likes with a price cheaper than 15: $\pi_{[\text{cname, rname}]}(\sigma_{[\text{price} < 15]}(\text{sells}) \bowtie \text{likes})$

The answer is the intersection of the two above.

$$\pi_{[\text{cname, rname}]}(\text{customer} \bowtie \text{restaurant}) \cap \pi_{[\text{cname, rname}]}(\sigma_{[\text{price} < 15]}(\text{sells}) \bowtie \text{likes})$$

(c) For all customer, find the pizza that they do not like (i.e., not in their likes table). The result should be the pair (cname, pizza).

Solution: Here, we know the pizza that the customer likes, which is the relation likes. To get the pairs not in here, we first need to get all possible pairs. All possible pairs can be obtained by a simple cross product.

$$(\pi_{[cname]}(\text{customer}) \times \text{pizza}) - \text{likes}$$