CS2102: Database Systems

Tutorial #1: *Relational Model & Relational Algebra*

Week 3 Guide

AY 2022/23 Sem 2

1 Discussions

The following questions are to be discussed during tutorial. All answers will be released with explanation.

1. **(Superkey)** Consider the following relation instance r of the relational schema $R(A, B, C, D)$.

R

A	B	C	D
0	0	0	1
2	1	2	0
1	1	2	0
0	0	1	2

   (a) Assuming that r is a *valid* relation instance of R, write down all the *possible* superkeys of R.

   (b) Additionally, suppose that it is also known that $\{A, C\}$ is *definitely* a superkey of R. Based on the additional information, write down all the *possible* candidate keys of R.

   (c) Which of these (*if any*) is likely be the candidate key of R?

Suggested Guide:

In this question, we are mainly *inferring* the properties of the relation schema from an instance instead of *enforcing* them. We start with the assumption that the relation instance r is valid, otherwise there are no solution to this.

We then look for *violations* by considering all subset of attributes of R and performing the check below. We denote the subset of attributes of interest as a and we let $b = R - a$ (*i.e.*, the set of attributes in R but not in a).

> If there are rows where all values in a are the same, but the values in b are not the same then a is not a superkey.

(a) The possible superkeys of R are $\{A, C\}$, $\{A, D\}$, $\{A, B, C\}$, $\{A, B, D\}$, $\{A, C, D\}$, and $\{A, B, C, D\}$. The are *possible* simply because there are no violations. If we have more entries in the database, some of these may turn out to be false.

Consider the set of attributes $\{B, C\}$. This is not a superkey because it does not uniquely identify $\{A, D\}$. There is a counter-example:

- Row 2: We have $B = 1$ and $C = 2$ with $A = 2$.
- Row 3: We have $B = 1$ and $C = 2$ with $A = 1$.

Hence, $\{B, C\}$ does not uniquely identify $\{A\}$.

(b) If $\{A, C\}$ is indeed a superkey of R, then $\{A, C\}$ must also be a candidate key of R since neither $\{A\}$ nor $\{C\}$ is a superkey of R (*based on* r). Furthermore, any superkey which is a superset of $\{A, C\}$ cannot be candidate key of R since they include $\{A, C\}$ and therefore not minimal.

The remaining two *possible* superkeys are $\{A, D\}$ and $\{A, B, D\}$. Both must be *possible* candidate keys of R since we do not know if $\{A, D\}$ is really a superkey of R or not. If $\{A, D\}$ is not a superkey then $\{A, B, D\}$ is a candidate key. Hence, we cannot discount the possibility that $\{A, B, D\}$ is a *possible* candidate key.

(c) Without further information (*e.g., the meaning of the attributes*), it is difficult to determine. But since we focus on A, C for some time, it can be easy to think that it might have some importance.

2. **(Foreign Key)** Consider a relational database consisting of two relations with schema $R(A, B)$ and $S(W, X, Y, Z)$ such that A is the primary key of R and W is the primary key of S (*for future use, we denote this with* $R(\underline{A}, B)$ *and* $S(\underline{W}, X, Y, Z)$ *where the attributes being underlined are parts of primary key*).

Let r and s be the current instances of R and S, respectively, as shown below.

R

A	B
3	0
2	1
1	1
0	0

S

W	X	Y	Z
0	4	0	NULL
1	NULL	2	NULL
2	1	2	NULL
3	0	1	NULL

Based on the current database instance above, write down all the *possible* foreign keys in S that refer to attribute A in R.

Suggested Guide:

Recap that the definition of *foreign key* requires only one of the following:

1. The value appears as *primary key* in the referenced relation.

2. The value is `NULL` (*or a tuple containing at least one* `NULL` *value*).

Similar to the previous question, we are *inferring* for possibilities. As such, we are looking for violations to discount the possibilities. Therefore, the *possible* foreign keys are W, Y, and Z.

3. **(Equivalent)** Recap that wwo queries Q_1 and Q_2 on a relational database with schema R are defined to be **(*strongly*) equivalent** (denoted by $Q_1 \equiv Q_2$) if for *every* valid instance r of R, either:

   - both Q_1 and Q_2 queries produces error, *or*
   - both Q_1 and Q_2 always compute the same results on r

   Consider a database with the following relational schema: $R(\underline{A}, C)$, $S(\underline{A}, D)$, and $T(\underline{X}, Y)$, with primary key attributes underlined. Assume all the attributes have integer domain. For each of the following pairs of queries Q_1 and Q_2, state whether or not $Q_1 \equiv Q_2$.

	$\mathbf{Q_1}$	$\mathbf{Q_2}$
(i)	$\pi_{[A]}(\sigma_{[A<10]}(R))$	$\sigma_{[A<10]}(\pi_{[A]}(R))$
(ii)	$\pi_{[A]}(\sigma_{[C<10]}(R))$	$\sigma_{[C<10]}(\pi_{[A]}(R))$
(iii)	$\pi_{[D,Y]}(S \times T)$	$\pi_{[D]}(S) \times \pi_{[Y]}(T)$
(iv)	$\pi_{[D,Y]}(S \times T)$	$\pi_{[D,Y]}(T \times S)$

Suggested Guide:

The reasoning given below is informal but for our purpose in this module, it is sufficient. Note that the order of the result does not matter in relational algebra because there is no ordering operator.

In practice, you may want to try to find *counter-example* first. If you cannot find a counter-example after some time, you may start using some algebraic properties to reason about it.

(i) **Equivalent**
This is because it does not matter if you do projection on A first or selection on A first, the result will have all rows where $A \geq 10$ removed *and* only contains attributes A.

(ii) **Not Equivalent**
This is because the second relational algebra is *invalid* since the selection condition refers to a non-existent attribute. Even if it is not invalid, then the condition is *trivially* satisfied. We choose to make this invalid instead.

(iii) **Equivalent**
The result will have the same attributes $\{D, Y\}$ in the exact same order (*i.e.*., (D, Y)). Furthermore, in both cases, for every element in S, it will be paired with an element in T. After removing duplicates (*either after Cartesian product as in* Q_1 *or after Cartesian product as in* Q_2), the remaining elements will be equivalent.

(iv) **Equivalent**
The intermediate result is *not equivalent* (*i.e.*, $S \times T \not\equiv T \times S$), but after rearranging the attributes, they will be equivalent.

4. **(Algebra)** Consider the following schema:

Relation	Description
$Pizzas(\underline{pizza})$	All the pizzas of interest.
$Customers(\underline{cname}, area)$	The name and location of each customer.
$Restaurants(\underline{rname}, area)$	The name and location of each restaurant.
$Recipes(\underline{pizza, ingredients})$	The ingredients used in each pizza.
$Sells(\underline{rname, pizza}, price)$	Pizzas sold by restaurants and the prices.
$Likes(\underline{cname, pizza})$	Pizzas that customers like.

Additionally, we have the following foreign key constraints on the database schema:

- $(Recipes.pizza) \rightsquigarrow (Pizzas.pizza)$
- $(Sells.rname) \rightsquigarrow (Restaurants.rname)$
- $(Sells.pizza) \rightsquigarrow (Pizzas.pizza)$
- $(Likes.cname) \rightsquigarrow (Customers.cname)$
- $(Likes.pizza) \rightsquigarrow (Pizzas.pizza)$

Answer each of the following queries using relational algebra. For those interested, you can try writing the relational algebra query on https://www.comp.nus.edu.sg/~cs2102/Tools/RelAlgebra/. The syntax can be found in https://www.comp.nus.edu.sg/~cs2102/Tools/RelAlgebra/help.html.

You may find the sample data on the file `T01.tbl`.

(i) Find all pizzas that Moe likes but is not liked by Lisa[1].

(ii) 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.

(iii) Suppose the relation *Likes* contains all information about all customers. In other words, if the pair $(cname, pizza)$ is not in the relation *Likes*, it means that the customer *cname* *dislikes* the pizza *pizza*. Write a relational algebra expression to find for all customers, the pizza that they dislike. The result should be of the form $(cname, pizza)$.

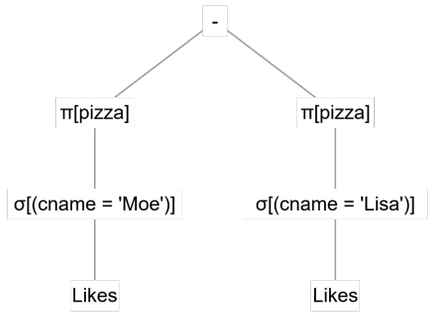
Suggested Guide:

Due to the complexity of some of the answers, we split the answers into subqueries. The final answer is always given as Q_{ans}.

(i) Let Q_1 be all pizza that Moe likes and Q_2 be all pizza that Lisa likes. Then Q_{ans} is the difference.

$$Q_1 := \pi_{\texttt{[pizza]}}(\sigma_{\texttt{[cname='Moe']}}(\texttt{Likes}))$$
$$Q_2 := \pi_{\texttt{[pizza]}}(\sigma_{\texttt{[cname='Lisa']}}(\texttt{Likes}))$$
$$Q_{ans} := Q_1 - Q_2$$

[1]The intention is to state it as *"all pizzas that Moe likes but Lisa does not like"*, however, this may indicate dislike which we have not discussed the underlying assumption yet.



NOTE:
The following answer is incorrect:

$$\begin{aligned} Q_1 &:= \texttt{Likes} \times \rho_{[\texttt{cname2}\leftarrow\texttt{cname},\texttt{pizza2}\leftarrow\texttt{pizza}]}(\texttt{Likes}) \\ Q_2 &:= \sigma_{(\texttt{[pizza=pizza2]})\wedge(\texttt{[cname='Moe']})\wedge(\texttt{[cname2<>'Bob']})}(Q_1) \\ Q_{ans} &:= \pi_{\texttt{[pizza]}}(Q_2) \end{aligned}$$

The above answer will compute exactly the set of pizzas that Alice likes *independent of whether Bob likes them*. This is because for each pizza p that Alice likes, we have $(\texttt{'Alice'}, p, \texttt{'Alice'}, p) \in \texttt{Likes} \times \texttt{Likes}$.

(ii) *Solution 1:* We join `Restaurants`, `Sells`, `Likes` and `Customers` tables since we need all these information. Then, we equate the restaurant names, customer names, pizza and the area.

$$\begin{aligned} Q_1 &:= \texttt{Customers} \times \rho_{[\texttt{rname2}\leftarrow\texttt{rname},\texttt{area2}\leftarrow\texttt{area}]}(\texttt{Restaurants}) \\ &\quad \times \rho_{[\texttt{rname3}\leftarrow\texttt{rname},\texttt{pizza3}\leftarrow\texttt{pizza}]}(\texttt{Sells}) \\ &\quad \times \rho_{[\texttt{cname4}\leftarrow\texttt{cname},\texttt{pizza4}\leftarrow\texttt{pizza}]}(\texttt{Likes}) \\ Q_2 &:= \sigma_{[(\texttt{area=area2})\wedge(\texttt{rname2=rname3})\wedge(\texttt{cname=cname4})\wedge(\texttt{pizza3=pizza4})]}(Q_1) \\ Q_{ans} &:= \pi_{[\texttt{cname},\texttt{rname}]}(Q_2) \end{aligned}$$

Solution 2: Solution 1 is actually equivalent to natural join of all the tables involved.

$$Q_{ans} = \pi_{[\texttt{cname},\texttt{rname}]}(\texttt{Customers} \bowtie \texttt{Restaurants} \bowtie \texttt{Sells} \bowtie \texttt{Likes})$$

Solution 3: Another way to think about this is to decompose the problem into two components. Q_1 is the table consisting of all customers and restaurants in the same area. Q_2 is the table consisting of all customers and restaurants such that the restaurant sells some pizza that the customer likes. The answer is the intersection of these two tables.

$$\begin{aligned} Q_1 &:= \pi_{[\texttt{cname},\texttt{rname}]}(\texttt{Customers} \bowtie \texttt{Restaurants}) \\ Q_2 &:= \pi_{[\texttt{cname},\texttt{rname}]}(\texttt{Sells} \bowtie \texttt{Likes}) \\ Q_{ans} &:= Q_1 \cap Q_2 \end{aligned}$$

(iii) Let Q_1 be all customer-pizza pairs (*i.e.*, all possible (C, P)). Then the final answer is $Q_1 - \texttt{Likes}$.

$$\begin{aligned} Q_1 &:= \pi_{\texttt{[cname]}}(\texttt{Customers}) \times \texttt{Pizzas} \\ Q_{ans} &:= Q_1 - \texttt{Likes} \end{aligned}$$

2 Challenge

The answers to the following questions is given without explanation. Please discuss them on Canvas.

1. **(Equivalent)** Recap that wwo queries Q_1 and Q_2 on a relational database with schema R are defined to be **(*strongly*) equivalent** (denoted by $Q_1 \equiv Q_2$) if for *every* valid instance r of R, either:

   - both Q_1 and Q_2 queries produces error, *or*
   - both Q_1 and Q_2 always compute the same results on r

   Consider a database with the following relational schema: $R(\underline{A}, C)$, $S(\underline{A}, D)$, and $T(\underline{X}, Y)$, with primary key attributes underlined. Assume all the attributes have integer domain. For each of the following pairs of queries Q_1 and Q_2, state whether or not $Q_1 \equiv Q_2$.

	$\mathbf{Q_1}$	$\mathbf{Q_2}$
(i)	$(R \times \pi_{[D]}(S)) \times T$	$R \times (\pi_{[D]}(S) \times T)$
(ii)	$\pi_{[A]}(R \cup S)$	$\pi_{[A]}(R) \cup \pi_{[A]}(S)$
(iii)	$\pi_{[A]}(R - S)$	$\pi_{[A]}(R) - \pi_{[A]}(S)$

Suggested Guide:

(i) **Equivalent:** Cartesian product is *associative*.

(ii) **Equivalent:** Because R and S are *union-compatible*.

(iii) **Not Equivalent:** Counter-example

R

A	B
10	10

S

C	D
10	20

2. **(Algebra)** Consider the following schema:

Relation	Description
$Pizzas(\underline{pizza})$	All the pizzas of interest.
$Customers(\underline{cname}, area)$	The name and location of each customer.
$Restaurants(\underline{rname}, area)$	The name and location of each restaurant.
$Recipes(\underline{pizza, ingredients})$	The ingredients used in each pizza.
$Sells(\underline{rname, pizza}, price)$	Pizzas sold by restaurants and the prices.
$Likes(\underline{cname, pizza})$	Pizzas that customers like.

Additionally, we have the following foreign key constraints on the database schema:

- $(Recipes.pizza) \rightsquigarrow (Pizzas.pizza)$
- $(Sells.rname) \rightsquigarrow (Restaurants.rname)$
- $(Sells.pizza) \rightsquigarrow (Pizzas.pizza)$
- $(Likes.cname) \rightsquigarrow (Customers.cname)$
- $(Likes.pizza) \rightsquigarrow (Pizzas.pizza)$

Answer each of the following queries using relational algebra. For those interested, you can try writing the relational algebra query on https://www.comp.nus.edu.sg/~cs2102/Tools/RelAlgebra/. The syntax can be found in https://www.comp.nus.edu.sg/~cs2102/Tools/RelAlgebra/help.html.

You may find the sample data on the file `T01.tbl`.

(i) For each restaurant, find the price of the most expensive pizzas sold by that restaurant. Exclude restaurants that do not sell any pizza.

(ii) Find all customer-pizza pairs (C, P) where the pizza P sold by some restaurant that is located in the same area as that of the customer C. Include customers whose associated set of pizzas is empty.

Suggested Guide:

(i) We split the answer into multiple subqueries.

$$Q_1 := \texttt{Sells} \times \rho_{[\texttt{rname2} \leftarrow \texttt{rname}, \texttt{price2} \leftarrow \texttt{price}, \texttt{pizza2} \leftarrow \texttt{pizza}]}(\texttt{Sells})$$
$$S_1 := \pi_{[\texttt{rname}, \texttt{price}]}(\sigma_{[(\texttt{rname}=\texttt{rname2}) \wedge (\texttt{price}<\texttt{price2})]}(Q_1))$$
$$S_2 := \pi_{[\texttt{rname}, \texttt{price}]}(\texttt{Sells}) - S_1$$

Here, $Q_{ans} = S_2$.

(ii) To include customers whose associated set of pizzas is empty, we will need *outer joins*. Here we use natural outer joins since we omit the conditions.

$$\pi_{[\texttt{cname}, \texttt{pizza}]}(\texttt{Customers} \rtimes (\texttt{Restaurants} \bowtie \texttt{Sells}))$$

3. **(Understanding RA)** Consider the following relational algebra query expressed on the database schema in Question 4.

$$R_1 := \pi_{[\texttt{pizza}]}(\sigma_{[\texttt{cname='Maggie'}]}(\texttt{Likes}))$$
$$R_2 := \pi_{[\texttt{rname}]}(\texttt{Sells}) \times R_1$$
$$R_3 := \pi_{[\texttt{rname}]}(R_2 - \pi_{[\texttt{rname}, \texttt{pizza}]}(\texttt{Sells}))$$
$$R_4 := \pi_{[\texttt{rname}]}(\texttt{Sells}) - R_3$$
$$R_5 := \pi_{[\texttt{pizza}]}(\sigma_{[\texttt{cname='Ralph'}]}(\texttt{Likes}))$$
$$R_6 := \pi_{[\texttt{rname}]}(\sigma_{[\texttt{pizza5}=\texttt{pizza}]}((\texttt{Sells} \times \rho_{[\texttt{pizza5} \leftarrow \texttt{pizza}]}(R_5))))$$
$$R_7 := R_4 - R_6$$

For each of the relational algebra expression R_i, write down a concise English sentence to precisely describe the information retrieved by R_i.

Suggested Guide:

R_1: The set of pizzas that Maggie likes.

R_2: The set of restaurant-pizza pairs (r, p) where r is a restaurant that sells some pizza and p is some pizza that Maggie likes. (**Note:** *r is not the restaurant that sells some pizza that Maggie likes*)

R_3: The set of restaurants that don't sell some pizza that Maggie likes.

R_4: The set of restaurants that sell all the pizzas that Maggie likes.

R_5: The set of pizzas that Ralph likes.

R_6: The set of restaurants that sells some pizza that Ralph likes.

R_7: The set of restaurants that sell all the pizzas that Maggie likes and don't sell any pizza that Ralph likes.