CSC343 A1

Haider Sajjad, Muskan Patpatia

June 8, 2019

Relations

- Restaurant(name, owner, capacity, country)
- Patron(PID, fname, lname, birthday)
- Dish(DID, name, dietary)
- Reservation(RID, PID, rname, date)
- Order(RID, DID, number)
- Rating(PID, rname, rating)

Part 1

1. Question: Report the name of the Patron that has given the highest rating to a restaurant. If there are ties, report all of them.

   Let $r1$ and $r2$ be two Rating relations

$$r1 \coloneqq \rho \text{ r1 Rating} \tag{1}$$
$$r2 \coloneqq \rho \text{ r2 Rating} \tag{2}$$

   Let nottop be the relation of all PID's that are not the maximum

$$nottop(PID) \coloneqq \Pi_{(r1.PID)} \ (\sigma_{r2.rating > r1.rating \wedge r2.PID \neq r1.PID} \ (r1 \times r2)) \tag{3}$$

   Let temp be the relation of all maximums, pid's that are in Rating but not in nottop. Natural Join temp with patron to get the names of the Patrons with pids in temp.

$$temp(PID) \coloneqq (\Pi_{PID} Rating) - nottop \tag{4}$$
$$\Pi_{fname,lname} \ (temp \bowtie Patron) \tag{5}$$

2. Question: Report the name of the restaurant for which the highest number of reservations were made. If there are ties report all of them.

   Can't be done with relational algebra operations.

3. Question: Report the PID(s) of the Patrons(s) who reserved a spot at a restaurant, but did not order anything.

   Make an pidsOrdered relation that holds the pid of every patron who has made an order by natural joining Order with Reservation. Then subtract pidsOrdered from the set of all Reservation pids, this will give all the patron pids who have a reservation but have not ordered anything.

$$pidsOrdered(PID) \coloneqq \Pi_{PID} \ (Order \bowtie Reservation) \tag{6}$$
$$(\Pi_{PID} Reservation) - pidsOrdered \tag{7}$$

4. Question: Report the name(s) of the Patrons(s) who have made a reservation to the restaurant named 'Boston Pizza' and ordered 3 of a dish called 'Margherita Pizza'.

   Make a relation called pBP that holds the pid and rid of every patron who has made a reservation at 'Boston Pizza'. And a relation called mp that holds the did of a dish called 'Margherita Pizza'.

$$pBP(PID, RID) \coloneqq \Pi_{PID,RID} \ (\sigma_{rname=\text{'Boston Pizza'}}(Reservation)) \tag{8}$$
$$mp(DID) \coloneqq \Pi_{DID} \ (\sigma_{name=\text{'Margherita Pizza'}}(Dish)) \tag{9}$$

   Let temp be the relation holding every order from Boston pizza that has 3 of any dish. Then, natural join temp and mp to get all the orders where 'Margherita Pizza' was ordered 3 times. And natural join again with Patron to get the names of the patrons.

$$temp \coloneqq \sigma_{number=3}(pBP \bowtie Order)) \tag{10}$$
$$\Pi_{fname,lname}[(\Pi_{PID}(mp \bowtie temp)) \bowtie Patron] \tag{11}$$

5. Question: Report the owner of the restaurant with the highest average rating. If there are ties, report all of them.

   We can't find average in relational algebra. Need to use aggregate sql functions.

6. Question: Report the capacities of the restaurants from which patrons have so far only ordered foods with a 'gluten-free' dietary restriction.

   Make a relation called glutenfree that holds the did, of every dish that is gluten free. And make a relation called gfOrders that holds the rid of every patron who has ordered a gluten free patron.

$$glutenfree(DID) \coloneqq \Pi_{DID}(\sigma_{directory=\text{'gluten-free'}}(Dish)) \tag{12}$$
$$temp(RID) \coloneqq \Pi_{RID}(glutenfree \bowtie Order) \tag{13}$$

   Make another relation called restnames that holds the restaurant name of every restaurant that serves 'gluten-free food'. Do this by natural joining temp with Reservation. Finally cross restnames and Restaurant to get the capacity of every resteraunt serving 'gluten-free'.

$$restnames(rname) \coloneqq \Pi_{rname}(temp \bowtie Reservation) \tag{14}$$
$$\Pi_{capacity}(\sigma_{restnames.rname=Restaurant.name}(restnames \times Restaurant)) \tag{15}$$

7. Question: Report the restaurant owner for which the very earliest reservation out of all the reservations in the database was made. Report any ties.

   Make two relations, r1, r2 both copies of Reservation.

$$r1 \coloneqq \rho \text{ r1 Reservation} \tag{16}$$
$$r2 \coloneqq \rho \text{ r2 Reservation} \tag{17}$$

   Next create 3 relations, notbottom, bottom, and restname.

   'notbottom' is the rid of every reservation where there exists some other date in Reservation that is smaller than the current rid date.

   'bottom' is the set of smallest (earliest) dates, if there are multiple dates tied there will be multiple rid's.

   'restname' is a set of restaurant names that the rid's in the bottom relation visit.

$$notbottom(RID) \coloneqq \Pi_{r2.RID}\ (\sigma_{r1.date<r2.date\ \wedge\ r1.RID\neq r2.RID}(r1 \times r2)) \tag{18}$$
$$bottom(RID) \coloneqq (\Pi_{RID}\ Reservation) - notbottom \tag{19}$$
$$restname(rname) \coloneqq \Pi_{rname}\ (bottom \bowtie Restaurant) \tag{20}$$

   Finally project the owner of the restaurants where earliest reservations were made.

$$\Pi_{owner}(\sigma_{rname=name}(restname \times Restaurant)) \tag{21}$$

8. Question: Report the PID(s) of the Patrons who have made reservations to the restaurant named 'Red Lobster' on their birthday.

   Make a relation called RLpid which holds the pid and date of every reservation made to 'Red Lobster'. Natural join RLpid with Patron, and select every row where date = birthday, and project out the pid.

$$RLpid(PID, date) \coloneqq \Pi_{PID,date}\ (\sigma_{rname=\text{'}RedLobster\text{'}}Reservation) \tag{22}$$
$$\Pi_{PID}\ (\sigma_{date=birthday}(RLpid \bowtie Patron)) \tag{23}$$

9. Question: Consider all patrons that have made reservations to at least two different restaurants. For each of those patrons, report their name, and the names and ratings of all of the restaurants they went to (not ones they rated without actually going to).

   Make two relations p1 and p2, both copies of Reservation.

$$p1 \coloneqq \rho \text{ p1 Reservation} \tag{24}$$
$$p2 \coloneqq \rho \text{ p2 Reservation} \tag{25}$$

   Next make 3 relations, 'atleast2res', 'names', and 'pidrate'.

   'atleast2res' holds the pid and restaurant name of every patron who have made reservations to at least two different restaurants.

'names' holds the pid, and names of every patron in 'atleast2res'.

'pidrate' holds the pid, restaurant name, and rating given by every patron in 'atleast2res'.

$$\begin{aligned} atleast2res(PID, rname) \coloneqq \rho \text{ atleast2res } (\Pi_{p1.PID, p1.rname}\ [\sigma(p1.PID = p2.PID \quad (26)\\ \wedge\ p1.RID \neq p2.RID \quad (27)\\ \wedge\ p1.rname \neq p2.rname)(p1 \times p2)]) \quad (28)\end{aligned}$$

the ρ atleast2res at the beginning of the statement is to clarify that all the columns belong to 'atleast2res', and they will be referred to by atleast2res.PID not p1.PID, this adjustment was needed in RelaX, added just in case. 'pidrate' and 'names' also includes this precaution.

$$names(PID, fname, lname) \coloneqq \rho \text{ names } (\Pi_{a.PID, pt.fname, pt.lname}[\sigma_{a.PID = pt.PID}(\rho \text{ a atleast2res} \times \rho \text{ pt Patron})] \quad (29)$$

In 'pidrate' we cross 'atleast2res' and Rating, and select for every element in 'atleast2res' so we can get the rating given only if the patron has been to the restaurant. In the query I assign a to be 'atleast2res' and 'rat' to be Rating to make it shorter.

$$\begin{aligned} pidrate(PID, rname, rating) \coloneqq \rho \text{ pidrate } (\Pi(a.PID, rat.rname, rat.rating) \quad (30)\\ [\sigma(a.PID = rat.PID \quad (31)\\ \wedge\ a.rname = rat.name) \quad (32)\\ (\rho \text{ a atleast2res} \times \rho \text{ rat Rating})] \quad (33)\end{aligned}$$

Finally cross pidrate, and names and project patron names, restaurant names, and restaurant ratings as required.

$$\begin{aligned} \Pi_{(n.fname, n.lname, p.rname, p.rating)} \quad (34)\\ [\sigma_{n.PID = p.PID}(\rho \text{ p pidrate} \times \rho \text{ n names})] \quad (35)\end{aligned}$$

10. Question: Report the name of all Restaurants that had reservations made on every day that someone made a reservation at the restaurant named 'Pickle Barrel'.
PickleBarrelDates stores a column of all the dates on which reservations were made to the resterunt named Pickel Barrel.

$$PickleBarrelDates(date) \coloneqq \Pi_{date}(\sigma_{rname='PickleBarrel'}(Reservation)) \quad (36)$$

AllReservations holds all the resteraunt names in the rname column and the respective dates in the date column on which the reservation at the respective. resteraunt was made.

$$AllReservations(rname, date) \coloneqq \Pi_{rname, date}(Reservation) \quad (37)$$

ShouldHaveBeen stores tuples of all the possible combinations that could have occurred.

$$ShouldHaveBeen(rname, date) \coloneqq (\Pi_{rname} AllReservations) \times PickleBarrelDates \quad (38)$$

Subtracting AllReservations which contains the tuple of all the restaurants and the dates on which reservations were made at these restaurants from the ShouldHaveBeen table which contains the tuple of all possible combinations of restaurants and the dates for the Pickle Barrel resteraunts will give us a table named WereNotAlways whoch contains the resteraunt names whose reservation dates did not match every date of reservation for Pickle Barrel. This includes the failures.

$$WereNotAlways(rname, date) \coloneqq ShouldHaveBeen - AllReservations \tag{39}$$

Subtracting the rname column of WereNotAlways table which contains the restaurants which failed the criteria from the rname column of AllReservations which contains all the resteraunts at which reservations were made will give us the restaurants that pass the criteria i.e. which had reservations made on every day that someone made reservation at Pickle Barrel.

$$WereAlways(rname) \coloneqq (\Pi_{rname} AllReservations) - (\Pi_{rname} WereNotAlways) \tag{40}$$

$$WereAlways \tag{41}$$

Part 2: Integrity constraints

1. Question: A restaurant owner can only own one restaurant.

   Let $r1$ and $r2$ be two Restaurant relations

$$r1 \coloneqq \rho \text{ r1 Restaurant} \tag{42}$$

$$r2 \coloneqq \rho \text{ r2 Restaurant} \tag{43}$$

   A restaurant owner can only own one restaurant is the same as saying a restaurant owner cannot own more than 1 restaurant.

$$\sigma_{r1.owner=r2.owner \,\wedge\, r2.name \neq r1.name} \; (r1 \times r2)) = \emptyset \tag{44}$$

2. Question: Patrons who did not make a reservation for a restaurant cannot review it.

   Let res be a Reservation relation, and rat be a Rating relation

$$res \coloneqq \rho \text{ res Reservation} \tag{45}$$

$$rat \coloneqq \rho \text{ rat Rating} \tag{46}$$

   Get every patron id who rated a restaurant along with the restaurant they rated and store that in temp. Check if there exists a pid, restaurant combination that does not exist in reservation, set to null.

$$temp(PID, rname) \coloneqq \Pi_{PID,rname} \; rat \tag{47}$$

$$temp - (\sigma_{PID,rname} Reservation) = \emptyset \tag{48}$$

3. Question: A Patron cannot make multiple reservations in one day for a restaurant that has a capacity less than 100.

   Let $res1$ and $res2$ be two Reservation relations

$$res1 \coloneqq \rho \text{ res1 Reservation} \tag{49}$$

$$res2 \coloneqq \rho \text{ res2 Reservation} \tag{50}$$

Make a relation 'temp', that holds the pid and rname of every row tuple where the patron has two reservations for the same restaurant on the same day.

$$\begin{aligned}
temp(PID, name) &\coloneqq \Pi_{res1.PID, res1.rname} && (51)\\
&(\sigma(res1.PID = res2.PID && (52)\\
&\wedge res1.rname = res2.rname && (53)\\
&\wedge res1.RID \neq res2.RID && (54)\\
&\wedge res1.date = res2.date)(res1 \times res2)) && (55)
\end{aligned}$$

Natural join temp with Restaurant over name (restaurant name), and get which restaurants have less than 100 capacity.

$$rest \coloneqq \rho \text{ rest Restaurant} \qquad (56)$$

$$\sigma_{rest.capacity<100}(rest \bowtie temp) = \emptyset \qquad (57)$$