Introduction to Databases

Tutorial 3

Dr Paolo Guagliardo

Fall 2020 (week 5)

Problem 1. Consider a schema with the following tables:

Customer over attributes *ID*, *Name*, *City*

Account over attributes *Number*, *Branch*, *CustID* and *Balance*

under the following constraints:

- no two rows of Customer have the same value for *ID*,
- no two rows of Account have the same value for *Number*, and
- all values in column *CustID* in Account appear in column *ID* in Customer.

Write the following queries in SQL:

(1) "ID and name of customers living in London who do **not** own any account in Edinburgh."

(2) "ID and name of customers who own an account in **every** branch."

(3) "ID and name of customers who own an account with a balance which is no less than the balance of any other account."

(4) "Customers who own an account with a balance that is at least 500 pounds higher than the average balance of all accounts in the same branch (of the account in question). Return the customer's ID, their name, and the corresponding account number."

Solution. The different alternatives written here for each query return precisely the same answers when no two rows in Customer have the same ID and there are no nulls in the database.

(1) One way is:

```sql
SELECT C.id, C.name
FROM   Customer C
WHERE  C.city = 'London'
  AND  C.id NOT IN ( SELECT A.custid
                     FROM   Account A
                     WHERE  A.branch = 'Edinburgh');
```

Another way is:

```sql
SELECT C.id, C.name
FROM   Customer C
WHERE  C.city = 'London'
  AND  NOT EXISTS ( SELECT *
                    FROM   Account A
                    WHERE  A.branch = 'Edinburgh'
                      AND  A.custid = C.id);
```

Yet another solution is:

```
SELECT C.id, C.name
FROM   Customer C
WHERE  C.city = 'London'
EXCEPT
SELECT C.id, C.name
FROM   Customer C JOIN Account A ON C.id = A.custid
WHERE  A.branch = 'Edinburgh';
```

(2) In relational calculus (see lecture slides) we can write:

$$\{ \ x, y \mid \exists z \ \textsc{Customer}(x, y, z) \wedge \forall u, w, v, k \ \textsc{Account}(u, w, v, k) \to \exists u', k' \ \textsc{Account}(u', w, x, k') \ \}$$

or, equivalently:

$$\{ \ x, y \mid \exists z \ \textsc{Customer}(x, y, z) \wedge \neg\exists u, w, v, k \ \textsc{Account}(u, w, v, k) \wedge \neg\exists u', k' \ \textsc{Account}(u', w, x, k') \ \}$$

This gives us the following SQL query:

```
SELECT C.id, C.name
FROM   Customer C
WHERE  NOT EXISTS ( SELECT *
                    FROM   Account A
                    WHERE  NOT EXISTS ( SELECT *
                                        FROM   Account A1
                                        WHERE  A1.branch = A.branch
                                          AND  A1.custid = C.id ));
```

The following is similar to the division-like RA expression we wrote in Tutorial 1 (awkward):

```
SELECT id, name
FROM   Customer
EXCEPT
SELECT T.id, T.name
FROM    ( SELECT C.id, C.name A.branch
          FROM   Customer C, Account A
          EXCEPT
          SELECT C.id, C.name, A.branch
          FROM   Customer C JOIN Account A ON C.id = A.custid ) T ;
```

Yet another way is by using aggregation:

```
SELECT   C.id, C.name
FROM     Customer C JOIN Account A ON A.custid = C.id
GROUP BY C.id, C.name
HAVING   COUNT(DISTINCT a.branch) = ( SELECT COUNT(DISTINCT branch)
                                      FROM   Account )
UNION
SELECT C.id, C.name
FROM   Customer C
WHERE  NOT EXISTS ( SELECT * FROM Account );
```

(3) Using aggregation:

```
SELECT DISTINCT C.id, C.name
FROM   Customer C JOIN Account A ON C.id = A.custid
WHERE  A.balance = ( SELECT MAX(balance)
                     FROM   Account );
```

Without aggregation (and similar to calculus query with $\forall$):

```
SELECT DISTINCT C.id, C.name
FROM   Customer C JOIN Account A ON C.id = A.custid
WHERE  A.balance >= ALL ( SELECT balance
                          FROM   Account );
```

Without aggregation (and similar to calculus query with $\neg\exists$):

```
SELECT DISTINCT C.id, C.name
FROM   Customer C JOIN Account A ON C.id = A.custid
WHERE  NOT EXISTS ( SELECT *
                    FROM   Account A1
                    WHERE  A1.balance > A.balance );
```

(4) We can write:

```
SELECT C.id, C.name, A.number
FROM   Customer C JOIN Account A ON C.id = A.custid
WHERE  A.balance - 500 >= ( SELECT AVG(A1.balance)
                            FROM   Account A1
                            WHERE  A1.branch = A.branch );
```

Problem 2 (optional). Given two relations R and S, each over attributes A, B (in this order), express the following relational calculus query in relational algebra:

$$\big\{x \mid \neg\big(\forall y\ R(x,y) \to S(x,y)\big) \land \neg\big(\exists z\ S(x,z) \land R(z,x)\big)\big\}$$

Use only the translation rules from RC to RA we have seen in class.

Solution. We first need to put the formula in the following form:

$$\underbrace{\big(\exists y\ R(x,y) \land \neg S(x,y)\big)}_{\varphi_1} \land \neg \underbrace{\big(\exists z\ S(x,z) \land R(z,x)\big)}_{\varphi_2}$$

Then we associate each variable with a distinct attribute name:

$$\{x \mapsto A, y \mapsto B, z \mapsto C\}$$

The translation of $R(x,y)$ is simply R and the translation of $S(x,y)$ is S. The translation of $\neg S(x,y)$ is $\mathsf{Adom}_A \times \mathsf{Adom}_B - S$. As $R(x,y)$ and $\neg S(x,y)$ have the same free variables, $R(x,y) \land \neg S(x,y)$ translates to $R \cap (\mathsf{Adom}_A \times \mathsf{Adom}_B - S)$. In turn, the translation of φ_1 is

$$\pi_A\big(R \cap (\mathsf{Adom}_A \times \mathsf{Adom}_B - S)\big) \tag{E_1}$$

The translation of $S(x,z)$ is $\rho_{B\to C}(S)$ and the translation of $R(z,x)$ is $\rho_{A\to C,\,B\to A}(R)$. As $S(x,z)$ and $R(z,x)$ have the same free variables, $S(x,z) \land R(z,x)$ translates to $\rho_{B\to C}(S) \cap \rho_{A\to C,\,B\to A}(R)$ and, in turn, the translation of φ_2 is

$$\pi_A\big(\rho_{B\to C}(S) \cap \rho_{A\to C,\,B\to A}(R)\big) \tag{E_2}$$

The translation of $\neg\varphi_2$ is $\mathsf{Adom}_A - E_2$. Since φ_1 and φ_2 have the same free variables, the final translation is $E_1 \cap (\mathsf{Adom}_A - E_2)$, which fully expanded gives us the following:

$$\pi_A\big(R \cap (\mathsf{Adom}_A \times \mathsf{Adom}_B - S)\big) \cap \Big(\mathsf{Adom}_A - \pi_A\big(\rho_{B\to C}(S) \cap \rho_{A\to C,\,B\to A}(R)\big)\Big)$$

Problem 3 (optional).

(a) Can we simplify the relational algebra expression obtained in Problem 2 into an equivalent expression that does not mention the active domain? If yes, give such an expression. Otherwise, explain why this is the case.

(b) How would you translate the relational calculus query of Problem 2 if the output tuple (i.e., the head of the query) were x, x rather than x?

Solution.

(a) Yes. Let E be the final expression obtained in Problem 2. The relations R and S (which are over attributes A, B) are both trivially subsets of $\mathsf{Adom}_A \times \mathsf{Adom}_B$. Then, $R \cap \big((\mathsf{Adom}_A \times \mathsf{Adom}_B) - S\big)$ is precisely $R - S$. So E can be simplified to

$$\underbrace{\pi_A(R - S)}_{E'} \cap \Big(\mathsf{Adom}_A - \underbrace{\pi_A\big(\rho_{B\to C}(S) \cap \rho_{A\to C,\, B\to A}(R)\big)}_{E''}\Big)$$

With a similar reasoning, since both E' and E'' (which are over attribute A) are subsets of Adom_A, we can simplify $E' \cap (\mathsf{Adom}_A - E'')$ to $E' - E''$. Thus, E is equivalent to

$$\pi_A(R - S) - \pi_A\big(\rho_{B\to C}(S) \cap \rho_{A\to C,\, B\to A}(R)\big)$$

(b) In general, if you have a relational calculus query with repetitions in the head, say $\{x, y, y, x, z, x \mid \psi\}$, it suffices to translate the body of the following equivalent query:

$$\big\{x, y, y', x', z, x'' \mid x = x' \wedge x = x'' \wedge y = y' \wedge \psi\big\}$$

For $\{x, x \mid \varphi\}$, where φ is the formula in the original query in Problem 2, we would have $\{x, x' \mid x = x' \wedge \varphi\}$. The translation of φ is the RA expression obtained in Problem 2, let us call it E, so if we map x' to attribute A', we obtain:

$$\sigma_{A=A'}(\mathsf{Adom}_A \times \mathsf{Adom}_{A'}) \cap (E \times \mathsf{Adom}_{A'})$$

The above expression is equivalent to $\sigma_{A=A'}\big(E \bowtie \rho_{A\to A'}(E)\big)$, but you are not required to rewrite it into this form.