Introduction to Databases
Advanced SQL

Dr Paolo Guagliardo

(v26.0)

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:

```
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:

```
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 RC we would write:

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

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 expression we wrote in RA (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 RC 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 RC 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 );
```