Databases (Sheet #2)

Marius Gavrilescu

1.
```
SELECT lname, fname, phone FROM Customerinfo
  NATURAL JOIN JobFor
  NATURAL JOIN WorkedOn
  NATURAL JOIN Employee
  WHERE ename = 'Smith';

SELECT employeeid FROM Employee e WHERE NOT EXISTS (
  SELECT * FROM (
    SELECT * FROM WorkedOn wo
      WHERE wo.employeeid = e.employeeid) AS x
    NATURAL JOIN JobFor
    NATURAL RIGHT JOIN Customerinfo
    WHERE customerid IS NULL);

SELECT customerid FROM Customerinfo ci
  WHERE fname = 'Louise'
  AND NOT EXISTS (
    SELECT * FROM JobFor jf
      NATURAL JOIN WorkedOn
      NATURAL JOIN Employee e
      WHERE jf.customerid = ci.customerid
      AND e.ename = 'Smith'
  );
```

2. $\pi_{lname,fname,phone}\,(\sigma_{ename=\text{"Smith"}}(\text{Employee}) \bowtie \text{WorkedOn} \bowtie \text{JobFor} \bowtie \text{Customerinfo})$

   $\pi_{employeeid}(\text{Employee}) - \pi_{employeeid}(\sigma_{customerid\ \text{IS NULL}}((\text{Employee} \bowtie \text{WorkedOn} \bowtie \text{JobFor}) \bowtie \text{Customerinfo}))$

   $\sigma_{fname=\text{"Louise"}}(\pi_{customerid}(\text{Customerinfo}) - \pi_{customerid}(\text{Customerinfo} \bowtie \text{JobFor} \bowtie \text{WorkedOn} \bowtie \sigma_{ename=\text{"Smith"}}(\text{Employee})))$

3.
```
SELECT customerid, count(jobid) AS njobs
  FROM JobFor
  GROUP BY customerid
  HAVING count(jobid) > 5;

SELECT customerid
  FROM JobFor
  NATURAL JOIN (
    SELECT * FROM WorkedOn
      WHERE employeeid = 253444) AS wo
  GROUP BY customerid
  HAVING count(jobid) > 5;

SELECT COUNT(*) FROM (
  SELECT salary, avg(salary) OVER () as avg_salary
    FROM Employee) AS tbl
  WHERE salary > avg_salary;
```

4. Let S be the set of all integers that appear in D, and $x \in \mathbb{Z} \setminus S$. Run the query for all combinations of variables where each variable is in $S \cup \{x\}$, and at least one of the variables is equal to x. If we get any results, the query is unsafe. Else the query is safe.

   The key observation here is that because we don't have inequalities, two variables that do not appear in the database cannot be distinguished one from another. Hence, if the query returns no results when we assign a variable to a value not in the database, then we know it returns no results when we assign a variable to any value not in the database. In contrast, if the query returns a result when we assign a variable to a value not in the database then we know it returns a result when we assign a variable to any value not in the database.

   If we introduce inequalities the algorithm fails because values not in the database are no longer identical. For example, if the database only contains the number 1 and we want to check $a > 1$, then the value 0 is not identical to the value 2.

   One way to fix this is to define an epsilon to be a number smallest than the smallest difference between two numbers in the database, and then to run the queries with values in the database and also with values that are epsilon greater or epsilon smaller than a value in the database.