CS2102: Database Systems

Mid Term

29 September 2020   12:30 - 14:00

Instructions

Submission Instructions

1. Please read **ALL** instructions carefully.

2. This assessment starts at 12:30 and ends at 14:00.

   - Submit your answers by 14:00.
   - No additional time will be given to submit.

3. This assessment consists of **ONE (1) problem** (*with SEVEN (7) sub-problems*) and comprises of **FOUR (4) printed pages**.

4. The maximum score of this assessment is **15 marks**.

5. Please type all your codes in the file provided in the template.

   - Do **NOT** change the filename of the code file.
   - Submit each code file in individual file (*i.e.*, not zipped).
   - Any change in filename may be penalized.
   - Unless specified otherwise, you are limited to **AT MOST 1 CTE**.
   - You are not allowed to declare other `VIEW` or use previously declared `VIEW`.

6. Please type all your other answers if possible.

   - If you are writing your answers, ensure that your handwriting is legible, or marks may be deducted.
   - For relational algebra, you may use the notation used in Coursemology or you may use (*i.e.*, copy-paste) the symbols available at http://dbis-uibk.github.io/relax/landing.

7. Submit **ALL** answer files (code/other) into your individual submission folder on ***LumiNUS***.

   - Submit each answer file in individual file (*i.e.*, not zipped).

8. Please make sure that the filename is unchanged as LumiNUS may change filename on duplicate submission (*e.g.*, on resubmission).

9. Only submissions made **before** deadline will be accepted.

   - Resubmission after deadline will not be accepted.
   - If resubmission causes filename changes, you may be penalized.

10. Failure to follow each of the instructions above may result in deduction of your marks.

Problem 1 : Table for Another Table

How powerful is a relational database? This midterm will explore the power of relational database by creating a series of tables that will be *simulating* another table except for the constraints. In other words, we will simulate only the content of any other table. In fact, it will simulate not only one table, but all possible tables at the same time. For simplicity, we will assume that:

1. All table contents are `VARCHAR(20)`.
2. All table names are `VARCHAR(20)`.
3. All table column names are `VARCHAR(20)`.

Consider a table `t1` created as follows:

```
CREATE TABLE t1 (
  a1  VARCHAR(20),
  a2  VARCHAR(20)
);
```

Assume that we have an instance with the following content:

t1	
a1	a2
10	100
20	200
30	300
40	400

If we further assume that the row ordering matters (*note: in SQL, the row ordering should not matter*), then we can assign each row with a *string equivalent* of a number starting from 0. Yes, this is a string, as will be made clear later because we will simulate the simulator table. Therefore, everything will be string (*or rather `VARCHAR(20)`*). We can then simulate the content of this table as a tuple ⟨ `tname`, `col`, `row`, `value` ⟩ where:

- `tname`:   is the name of the table being simulated.
- `col` :   is the column name of the table with the name `tname` being simulated.
- `row` :   is the string equivalent of row number (*starting with 0*) of the table with the name `tname` being simulated.
- `value`:   is the value of table with the name `tname` with column `col` at row `row` being simulated.

The table `t1` above can then be simulated with the following table called `simulator`:

simulator			
tname	col	row	value
t1	a1	0	10
t1	a2	0	100
t1	a1	1	20
t1	a2	1	200
t1	a1	2	30
t1	a2	2	300
t1	a1	3	40
t1	a1	3	400

In fact, the *schema* of the tables can also be simulated using the following table:

schema	
tname	col
t1	a1
t1	a2

NOTE: For any SQL query questions (*e.g.*, Part (d), Part (e), and Part (g)), you should check that your answer can be run on PostgreSQL.

a) [**1 mark**] What are the keys of the table `simulator` and the table `schema`? Remember, keys are minimal superkeys.

(tname, col, row)   **(tname, col)**

b) [**2 marks**] Note that the table `simulator` and the table `schema` are related since a table and its schema must also be related. What is/are the likely *foreign key* constraints on both the table `simulator` and the table `schema`. Express your answer(s) as a list the following statements:

> The attributes ⟨ `attr1`, `attr2`, ⋯ ⟩ of table `referencing_table` references the attributes ⟨ `att3`, `attr4`, ⋯ ⟩ of table `referenced_table`

The attributes <tname, col> of table simulator references the attributes <tname, col> of table schema

c) [**2 marks**] Consider the following table to be simulated:

t2		
b1	b2	b3
1	10	123
2	20	456

Created as follows:

```
CREATE TABLE t2 (
  b1  VARCHAR(20),
  b2  VARCHAR(20),
  b3  VARCHAR(20)
);
```

simulator
tname, col, row, value
t2, b1, 0, 1
t2, b2, 0, 10
t2, b3, 0, 123
t2, b1, 1, 2
t2, b2, 1, 20
t2, b2, 1, 456

schema
tname, col
t2, b1
t2, b2
t2, b3

What is the contents of the table `simulator` and table `schema` when we are simulating the table t2? Write **only** the contents of the table `simulator` and table `schema` that are *relevant* for table t2 (*i.e.*, should not contain content for other tables including t1).

d) [**2 marks**] Consider the following query on the *unsimulated table* `t1`:
`SELECT * FROM t1 WHERE a1 > '25';`
Yes, the condition is that `a1` is *greater than* a string. This is a normal string operation, *no need to cast into other type.*

Assuming the content of table `t1` *when not simulated* as above, will result in the following table:

result	
a1	a2
30	300
40	400

CREATE VIEW p1d (a1,a2) AS
select S1.value as a1, S2.value as a2
from simulator S1 join simulator S2 on S1.col < S2.col and S1.row = S2.row
where tname = 't1' and S1.value > '25';

Without using any CTE, write an *equivalent* query on the simulated table `simulator` that will produce exactly the same result as the given query. Your query should work for any valid content of *unsimulated tables* `t1` as well as table `simulator` which may be simulating other tables as well. You should assume that the *unsimulated version* of table `t1` does not exists and only being simulated (*i.e.*, `t1` cannot appear in `FROM` clause). Your query should begin with the following `CREATE VIEW` to simplify the result:
`CREATE VIEW p1d (a1,a2) AS`
which is also provided in the file `p1d.txt`

e) [**3 marks**] Consider the following query on *unsimulated tables* `t1` and `t2`:
`SELECT * FROM t1,t2 WHERE a1 = b2;`

Assuming the content of table `t1` and table `t2` *when not simulated* as above, will result in the following table:

```
CREATE VIEW p1d (a1,a2, b1, b2, b3) AS
select a1, a2, b1, b2, b3
from
(select S1.value as a1, S2.value as a2
from simulator S1 join simulator S2 on S1.col < S2.col and S1.row = S2.row
where tname = 't1') join
(select S3.value as b1, S4.value as b2, S5.value as b3
from (simulator S3 join simulator S4 on S3.col < S4.col and S3.row =
S4.row) join simulator S5 on S4.col < S5.col and S4.row = S5.row
where tname = 't2') on a1 = b2;
```

result				
a1	a2	b1	b2	b3
10	100	1	10	123
20	200	2	20	456

Without using any CTE, write an *equivalent* query on the simulated table `simulator` that will produce exactly the same result as the given query. Your query should work for any valid content of *unsimulated tables* `t1` and `t2` as well as table `simulator` which may be simulating other tables as well. You should assume that the *unsimulated version* of table `t1` and `t2` do not exists and only being simulated (*i.e.*, `t1` or `t2` cannot appear in `FROM` clause). Your query should begin with the following `CREATE VIEW` to simplify the result:

```
CREATE VIEW p1e (a1,a2,b1,b2,b3) AS
```

which is also provided in the file `p1e.txt`

f) [2 marks] Since `simulator` is simply another *table*, we can also simulate our `simulator` table. Let's call this `simulator2` (*for simulator level 2*). Obviously we can also simulate `simulator2`, and so on... However, we will stop at `simulator2` for now. What you should note is that the *schema* for `simulator2` is the same as `simulator`, except for the table name! Consider the following *unsimulated* table `t0`:

t0
a0
42
2102

simulator
tname, col, row, value
t0, a0, 0, 42
t0, a0, 1, 2102

Hopefully, if you can understand the parts above, you should know what is the content of `simulator` when simulating `t0`. Now, what we want you to do is to *simulate* the content of `simulator` when simulating `t0`.

simulator 2
tname, col, row, value
simulator, tname, 0, t0
simulator, col, 0, a0
simulator, row, 0
simulator, value, 0, 42
simulator, tname, 1, t0
simulator, col, 1, a0
simulator, row, 1, 1
simulator, value, 1, 2102

simulator2			
tname	col	row	value

The header is already given to you, all you have to do is fill in the values.

g) [3 marks] Consider the following query on *unsimulated table* `t0`:
`SELECT * FROM t0 WHERE a0 > '1010';` Yes, the `a0` is *greater than* a string. This is a normal string operation.

Assuming the content of table `t0` *when not simulated* as shown in part (f), will result in the following table (*note the string greater than is the usual interpretation of lexicographical ordering*):

t0
a0
42
2102

Write an *equivalent* query on the simulated table `simulator2` that will produce exactly the same result as the given query. Your query should work for any valid content of *unsimulated tables* `t0` and `simulator` (as it is being simulated) as well as table `simulator2` which may be simulating other tables as well. You should assume that the *unsimulated version* of table `t0` and `simulator` do not exists and only being simulated (*i.e.*, `t0` or `simulator` cannot appear in `FROM` clause). Your query should begin with the following `CREATE VIEW` to simplify the result:

```
CREATE VIEW p1g (a0) AS
```

which is also provided in the file `p1g.txt`

HINT: Since you are allowed to use at most 1 CTE, use it to compute `simulator` first!

```
with simulator as (
   select S1.value as tname, S2.value as col, S3.value as row, S4.value as value
   from (((simulator2 S1 join simulator2 S2 on S1.col = 'tname' and S2.col = 'col' and S1.row = S2.row)
join simulator2 S3) on S3.col = 'row' and S2.row = S3.row)) join simulator2 S4 on S4.col='value' and
S3.row = S4.row
   where tname='simulator'
)

select value as a0
from simulator
where tname='t0', col='a0' and value > '1010';
```