CSE 414 Midterm Exam
Winter 2020
February 7, 2020

- Write your name and UW student number below.
- This is a **closed-book exam.** You are allowed **one *handwritten* page** of notes, written on both sides.
- No electronic devices are allowed, including cell phones (even when used as watches). Silence your cell phones and place them in your bag.
- Solutions are graded on correctness and ***clarity***. Each problem has a relatively simple & straightforward solution. Partial solutions will be graded for partial credit.
- Please write your answers in the boxed space provided. Clearly mark your solutions. You may use blank sides as scratch paper. **Do not use additional scratch paper**.
- **Skip around if you're having difficulty**. Watch your time. For example, if you feel confident on transactions, you may want to solve txn problems first.
- *Relax. You are here to learn. Good luck!*

Relational algebra operators:

Union ∪	Difference \	Cartesian product x	Projection π
Rename ρ	Duplicate elimination δ	Grouping/aggregation ɣ	Join ⋈
Sorting τ	Left outer join ⟕	Intersection ∩	Selection σ

By writing your name below, you certify that you have not received unpermitted aid for this exam, and that you will not disclose the contents of the exam to anyone in the class who has not taken it.

NAME: _____

STUDENT NUMBER: _____

Problem	Points	Problem	Points	Problem	Points
1	20+	5	10	8	5
2	10	6	15	9	7
3	10	7	5	10	8
4	10			**Total**	**100+**

Part I: Job Search

Shana wants to find a company job. To find companies that better match her life goals, she consults a public database table:

`Facts(cid, name, dept, budget19, hiringCap, year, empLeft)`

- No primary key is declared for the `Facts` table
- All attributes of `Facts` have the `NOT NULL` constraint
- `cid` is an integer that uniquely identifies a company
- `name` is a company `cid`'s public name; a company has one name; multiple companies may have the same name
- `dept` is a company `cid`'s department name (e.g. *Marketing*; *Research*; *HR*) listed in the company's public records; a company has many departments; multiple companies can have the same department name
- `budget19` is how much money the company `cid` allocates to a `department` in the year 2019, specified in dollars; a company has one 2019 budget per department; multiple companies may have the same 2019 budget for a particular department
- `hiringCap` is the most employees that could be realistically hired into a company's department, due to minimum wage. It is calculated as `budget19 / $15,000`
- `year` is a year
- `empLeft` is how many employees left the company `cid` in the `year` listed

Here is the result of the SQL query `SELECT * FROM Facts LIMIT 6`:

```
cid   name       dept       budget19   hiringCap  year  empLeft
---   --------   ---------  --------   ---------  ----  -------
6     Costcom    Marketing  500,000    33.3       2018  300
6     Costcom    Marketing  500,000    33.3       2019  400
6     Costcom    Research   750,000    50         2018  300
6     Costcom    Research   750,000    50         2019  400
2     TinyFirm   HR          15,000     1         2018    1
2     TinyFirm   HR          15,000     1         2019    0
```

As an aspiring student of data management, you quickly criticize the public database's design and inform Shana that this public data could be organized with less redundancy.

TESTING CODE:

```
CREATE TABLE Facts(cid int, name text, dept text, budget19 int, hiringCap float, year int,
empLeft int);

INSERT INTO Facts VALUES(6, 'Costcom', 'Marketing', 500000, 33.3, 2018, 300);
INSERT INTO Facts VALUES(6, 'Costcom', 'Marketing', 500000, 33.3, 2019, 400);
INSERT INTO Facts VALUES(6, 'Costcom', 'Research', 750000, 50, 2018, 300);
INSERT INTO Facts VALUES(6, 'Costcom', 'Research', 750000, 50, 2019, 400);
INSERT INTO Facts VALUES(2, 'SmallFirm', 'HR', 15000, 1, 2018, 1);
INSERT INTO Facts VALUES(2, 'SmallFirm', 'HR', 15000, 1, 2019, 0);
INSERT INTO Facts VALUES(3, 'LargeFirm', 'HR', 6000000, 400, 2018, 5);
INSERT INTO Facts VALUES(3, 'LargeFirm', 'HR', 6000000, 400, 2019, 7);
```

1a) List functional dependencies present in the `Facts` table.

cid → name
cid, dept → budget19
budget19 → hiringCap
cid, year → empLeft

[10 points]
Found 1 FD ⇒ 3 points
Found 2 FD ⇒ 6 points
Found 3 FD ⇒ 8 points
Found 4 FD ⇒ 10 points
Found incorrect FD ⇒ -1 points per

1b) Decompose the `Facts` table into Boyce-Codd Normal Form (BCNF). Show your work! Circle the tables in your final normalized decomposition.

`Facts(cid, name, dept, budget19, hiringCap, year, empLeft)`

Decompose on cid → name
F1(cid, name), F2(cid, dept, budget19, hiringCap, year, empLeft)

Decompose on cid, dept → budget19, hiringCap [don't forget to take the closure]
F1(cid, name), F3(cid, dept, budget19, hiringCap), F4(cid, dept, year, empLeft)

Decompose on budget19 → hiringCap
F1(cid, name), F5(budget19, hiringCap), F6(cid, dept, budget19), F4(cid, dept, year, empLeft)

Decompose on cid, year → empLeft
F1(cid, name), F5(budget19, hiringCap), F6(cid, dept, budget19), F7(cid, year, empLeft), F8(cid, year, dept)

[10 points]
(note, 1b is graded independently of 1a; we use whatever FDs you wrote in 1a, even if they are wrong)
Lossy decomposition => -4
Incomplete decomposition => -4
Unclear => -2
Incorrect FD => -2

1c) [EXTRA CREDIT] Do additional redundancies exist in your BCNF tables? If no, argue why not. If yes, further refine your tables to eliminate additional redundancies (losslessly).

Yes, additional redundancies exist. A formal solution to this problem requires knowledge of multivalued dependencies. Informally, however, the solution can be found by simply noticing that the table F8(cid, year, dept) is completely redundant, as its information is present in the natural join of F6(cid, dept, budget19) and F7(cid, year, empLeft). An inner join is sufficient, since the `dept` and `year` attributes are `NOT NULL` in `Facts`.

In relation F8 there exist the multivalued dependencies
cid ↠ year
cid ↠ dept

These dependencies indicate that the years for which we have past empLeft data are independent of a company's departments. Also, a particular cid determines the set of years and the set of departments present for that company. In Fourth Normal Form (4NF), we eliminate these dependencies by decomposing F8(cid, year, dept) into
F9(cid, year), F10(cid, dept)

Additionally, we note that F9 = $\pi_{\text{cid, year}}$(F7) and F10 = $\pi_{\text{cid, dept}}$(F6).
Therefore, these tables can be completely eliminated.

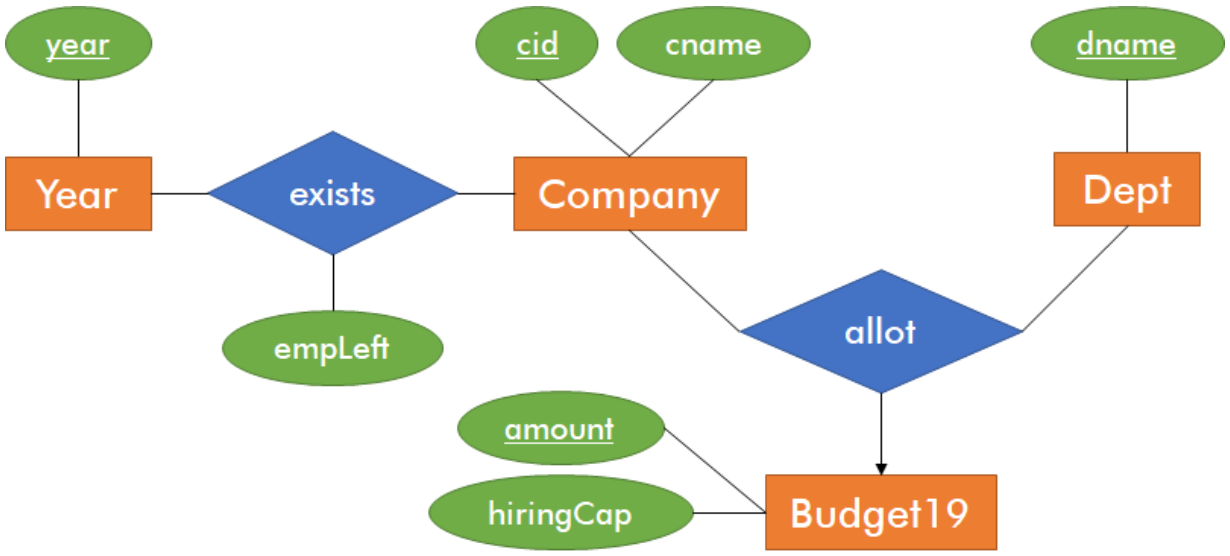
A full decomposition is
F1(cid, name), F5(budget19, hiringCap), F6(cid, dept, budget19), F7(cid, year, empLeft)

[5 extra points]

+1 for "Yes"
+2 for finding the correct table
+2 for good explanation

2) You contact the owners of the public database and offer them a better design. **Draw an E/R diagram that captures the semantics of the `Facts` table**. Remember: no attribute in an E/R diagram can be a set of values (i.e., do not violate the 1NF constraint).

```
Facts(cid, name, dept, budget19, hiringCap, year, empLeft)
```



[10 points]
Full credit for a “reasonable” design that does not violate 1NF.
2 Got all the attributes.
2 Separate entities (relationships)
2 There is an empLeft for each year. A company has many years.
2 A company has a budget19 for each of its departments. hiringCap with budget19.
2 Primary Keys
-1 or 2 per violate 1NF

`Facts(cid, name, dept, budget19, hiringCap, year, empLeft)`

3) High funding to HR departments and high employee retention (i.e., low employee turnover) seem like decent indicators of employee happiness. Doing the best you can with the public Facts database, you aim to show Shana a list of companies where employees may be happier.

Use only the original Facts table (do not use your BCNF decomposed tables).

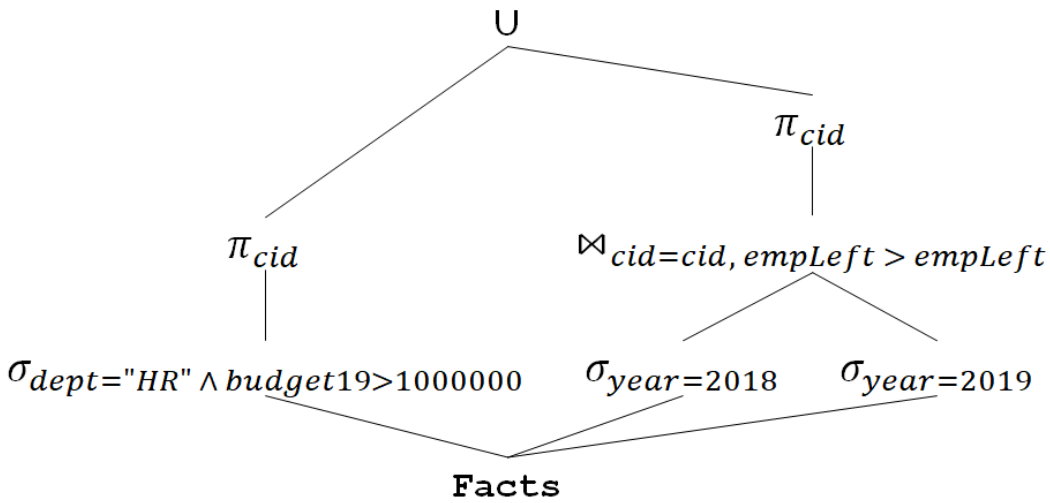
Either write a SQL query ***or*** draw a Relational Algebra plan (your choice) *that returns company ids (cids) whose 2019 funding to their HR department exceeds $1,000,000* ***OR*** *whose turnover (empLeft) has decreased from the year 2018 to the year 2019.*

You may assume that every company listed in `Facts` has an `empLeft` listed for both 2018 and 2019.

Your SQL query or RA plan should return a relation with a single attribute: **cid**. Though Shana appreciates solutions that don't output duplicate cids, duplicates are acceptable for this problem.

Facts(cid, name, dept, budget19, hiringCap, year, empLeft)

```
Select cid
From Facts
Where dept = 'HR' and budget19 > 1000000
UNION
Select A.cid
From (select cid, empLeft from Facts where year = 2018) A,
(select cid, empLeft from Facts where year = 2019) B
Where A.cid = B.cid and A.empLeft > B.empLeft;
```



[10 points]
3 for HR
3 for empLeft decrease
2 for union
2 for output schema cid

```
Facts(cid, name, dept, budget19, hiringCap, year, empLeft)
```

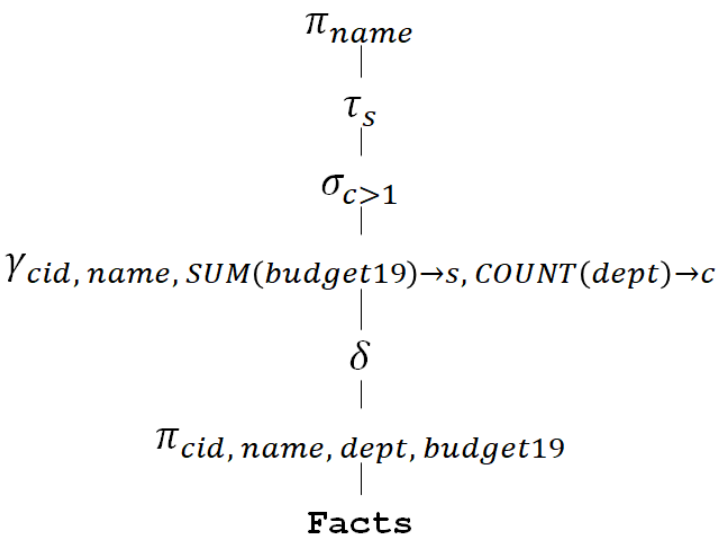
4) Shana wants to see the smallest companies first! But not the tiny single-department ones.

Use only the original Facts table (do not use your BCNF decomposed tables).

Either write a SQL query ***or*** draw a Relational Algebra plan (your choice) *that returns company names that have more than one department, ordered ascending by their total 2019 budget across all departments.*

Your SQL query or RA plan should return a relation with a single attribute: **name**. Duplicate names are allowed. Do not sum together the 2019 budgets of companies that have different cids and the same name.

```
SELECT x.name
FROM (SELECT DISTINCT cid, name, dept, budget19 FROM Facts) x
GROUP BY x.cid, x.name
HAVING COUNT(x.dept) > 1
ORDER BY SUM(x.budget19);
```



[10 points]
Subtract one point if the student uses Facts instead of the FROM subquery.
1 Correct output schema name
3 Only companies with >1 department
2 Order by budget19
2 Grouped on cid; Internally group on name
2 Subquery to deduplicate budget19

```
Facts(cid, name, dept, budget19, hiringCap, year, empLeft)
```

5) Knowing which department has the highest 2019 budget for each company could be helpful.

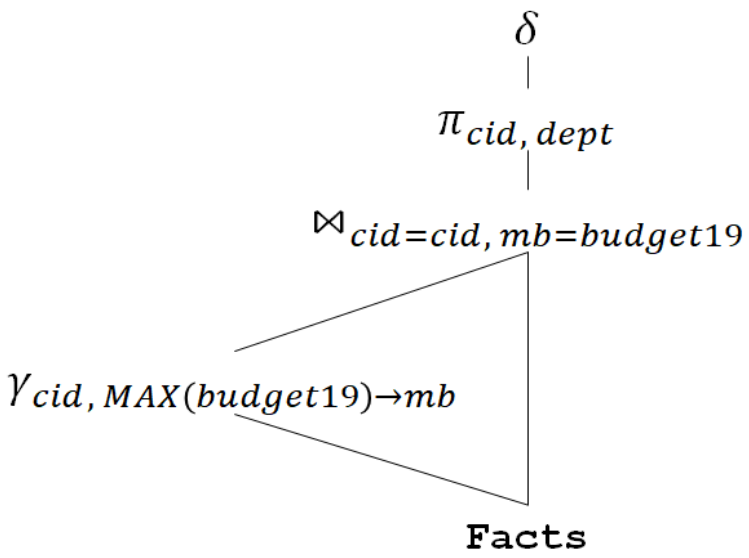
Use only the original Facts table (do not use your BCNF decomposed tables).

Either write a SQL query ***or*** draw a Relational Algebra plan (your choice) *that returns, for each company id, the department name whose 2019 budget is largest among all departments in that company.*

Your SQL query or RA plan should return a relation with the attributes: **cid, dept**. Do not output duplicates. If more than one department has the same largest 2019 budget in a particular company, output all their department names.

This is a classic witness problem.

With MaxBudget as (
Select cid, MAX(budget19) as mb
From Facts
Group by cid
)
Select DISTINCT f.cid, f.dept
From Facts f, MaxBudget m
Where f.cid = m.cid and f.budget19 = m.mb



[10 points]
1 DISTINCT
5 Finding the maximum budget19 for each company (subquery) with group by

2 Correct output schema
2 Outer query

Part II: Stop the Swine Flu!

The World Health Organization (WHO) is concerned about the possibility that a swine flu may grow to pandemic levels. Given the WHO's limited resources, they need your help to determine for which countries they should focus their efforts.

The WHO designed their database as follows:

```
CREATE TABLE Country(        -- Country data
     cname TEXT PRIMARY KEY, -- country name
     population INT,         -- country population
     cases INT);             -- # of confirmed swine flu cases
CREATE TABLE Member(         -- A country resides in a region
     cname TEXT PRIMARY KEY REFERENCES Country,
     rname TEXT REFERENCES Region);
CREATE TABLE Region(         -- Region data
     rname TEXT PRIMARY KEY, -- region name
     hospitals INT);         -- # of hospitals in region
```

6) The WHO plans to prioritize countries that ***either*** *(a) reside in a region with 0 hospitals* ***or*** *(b) have between 100 and 1100 confirmed swine flu cases*. It is known that at least one region has 0 hospitals.

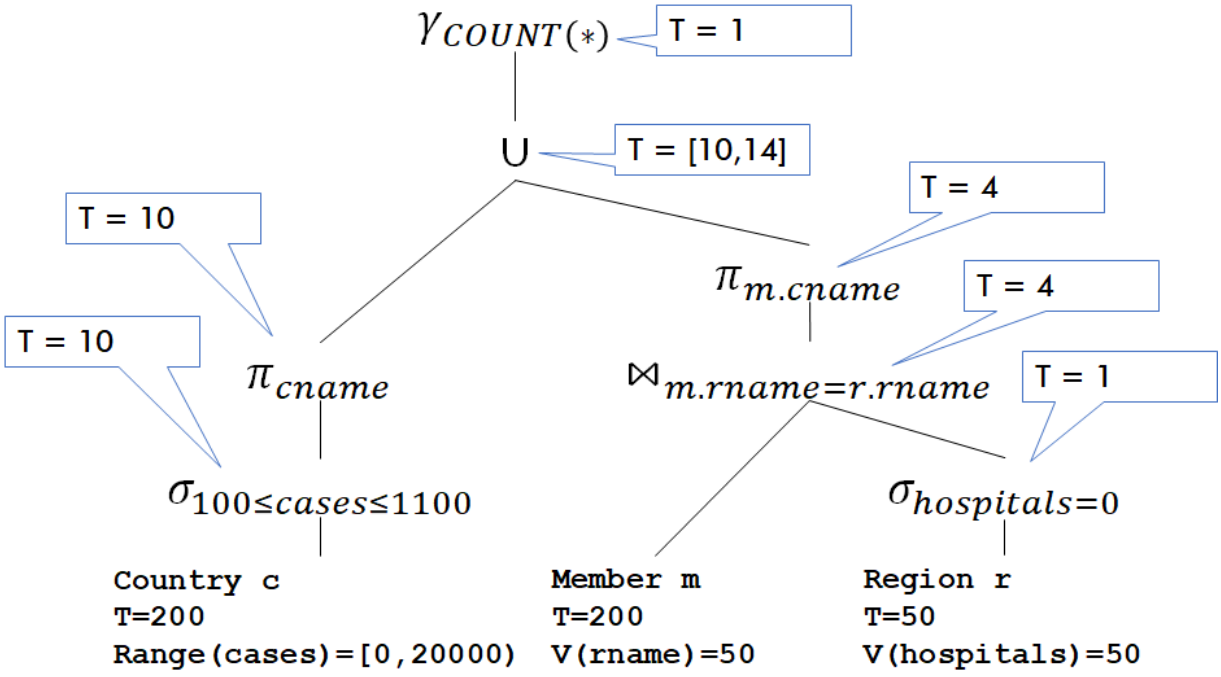
First the WHO writes a query to check the feasibility of their scheme: computing the number of countries that fit their criteria. The WHO discloses their master (relational algebra) plan to you, along with statistics on the relations in their database, on the next page.

Unfortunately, WHO ran out of Azure credits, so they can't compute an exact answer. You realize you can estimate an answer to their problem by estimating cardinalities.

Estimate the cardinality of each operator in the WHO's RA plan. Show your work!

Defend your answers by stating assumptions made (i.e., we can award full credit for unusual answers if backed up with reasonable assumptions or explanations).
If you wish, you are allowed to give answers as a range
[<minimum cardinality estimate>, <maximum cardinality estimate>].

Please write your answers on the next page, along with showing your work.



The cases calculation is (1100-100)/(20000-0) * 200 = (1/20) * 200 = 10. We assume that cases is uniformly distributed on its range [0,20000) across all countries. The calculation (1101-100)/(20000-0) is perhaps more accurate, but the math isn't nearly as nice.

Projection does not change the number of tuples.

Because hospitals is a key, selecting a single hospital value results in one tuple. The problem states that 0 is one of the unique values of hospitals.

The join operator uses the fact that m.rname is a foreign key to r.rname, alongside the fact that V(Member, rname) = T(Region), in order to guarantee that the 1 region present from Region exists in Member. We assume the countries in Member are uniformly distributed among the 50 regions.

The union operator results in 10 tuples at minimum (if the inputs fully overlap) and 14 tuples at maximum (if the inputs are disjoint).

Aggregation (without group by) results in 1 tuple.

[14 points]
2 points per blank
Points are given for correct calculations that use an incorrect input estimate.

Provide the WHO with your answer: ***How many countries do you estimate match the WHO's criteria***? You may give a single answer or a range of valid answers.

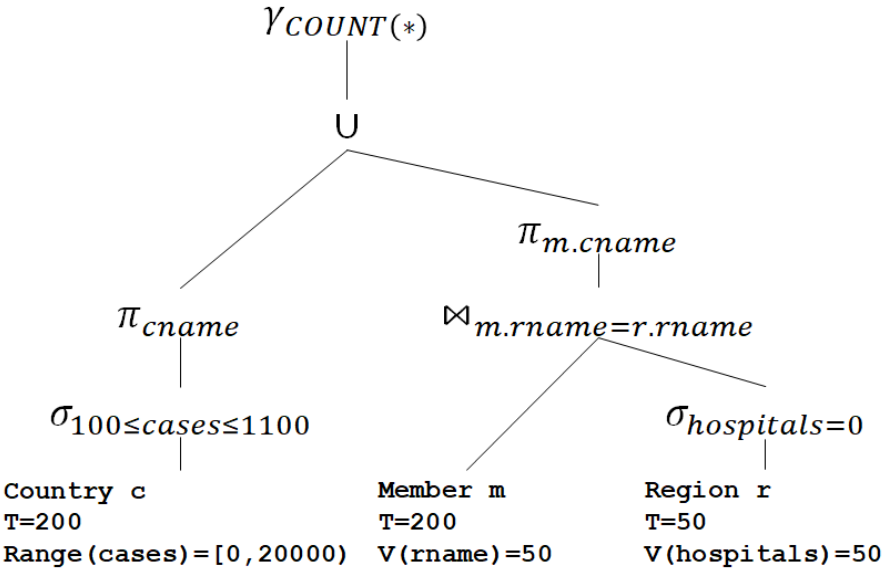
Between 10 and 14 countries. (This is the estimate right below the COUNT aggregate.)

We estimate 10 countries have between 100 and 1100 confirmed swine flu cases.
We estimate 4 countries have 0 hospitals in their region.
We do not know whether these estimates overlap or are disjoint.

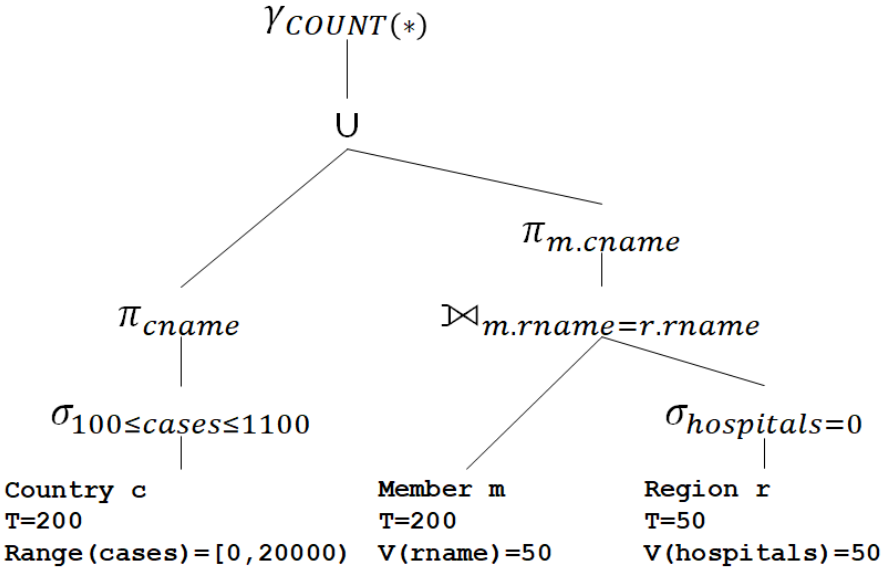
[1 points]

7) An earthquake strikes and jumbles the WHO's RA plan! The join gets switched into a left outer join:

Original:



Jumbled:

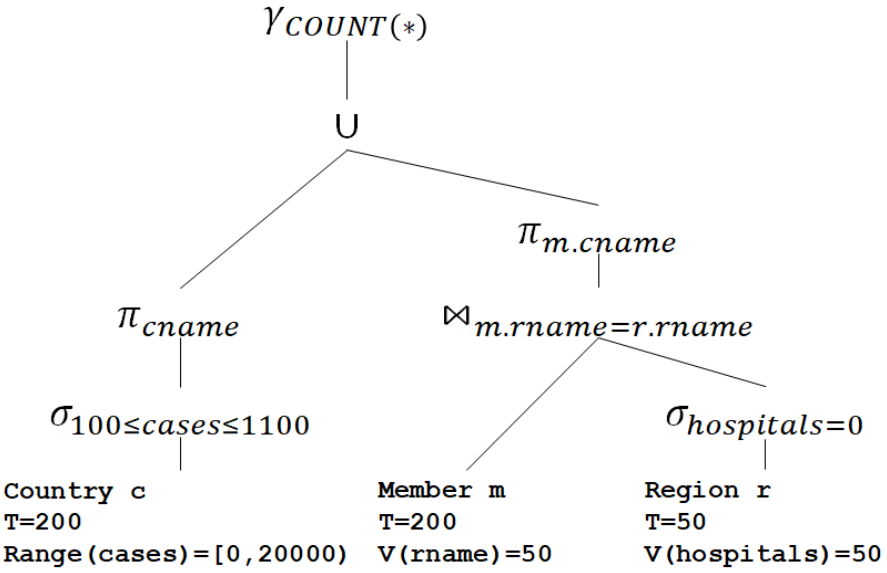


Is this jumbled plan equivalent to the original query plan? State yes or no with a brief explanation.

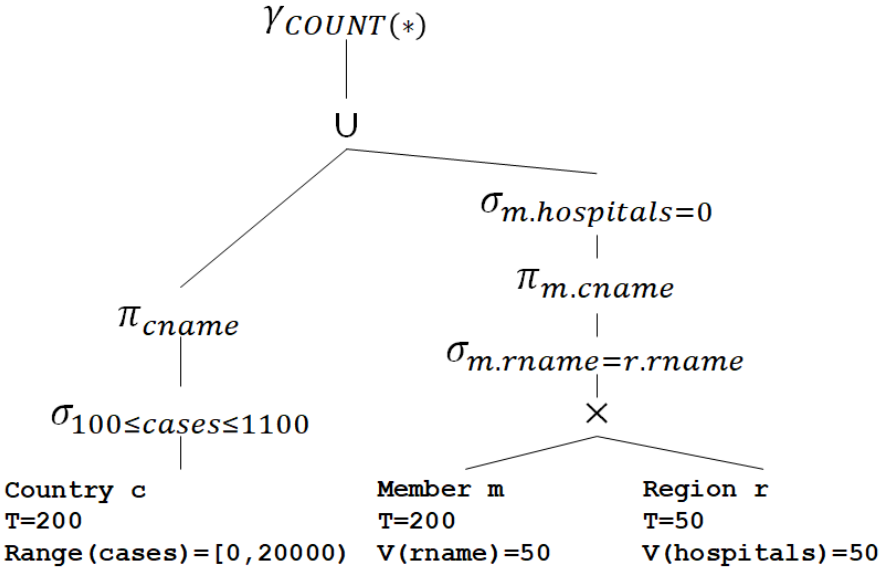
No, the outer join is not equivalent to an inner join, because the hospitals selection can filter out regions so that some Member countries have no match, meaning they pair with nulls. [5 points]

8) As soon as the WHO restores their original plan, another earthquake strikes! Now the join is lost completely, a whirlwind around it and the surrounding operators:

Original:



Jumbled:



Is this jumbled plan equivalent to the original query plan? State yes or no with a brief explanation.

No. The join is correctly decomposed into a cartesian product followed by a selection of the join condition, but the projection onto cname runs before the selection of hospitals = 0. Therefore, hospitals is no longer in the schema of its input, rendering the RA plan invalid.

[5 points]

Part III: Concurrent Flu

The swine flu is spreading concurrently! Help the WHO track the flu's progress by updating their database from around the world as fast as possible.

The following two transactions need to run:

Txn 1:
1a) x = SELECT cases FROM Country WHERE cname = 'France';
1b) x = x + 200;
1c) UPDATE Country SET cases = `x` WHERE cname = 'France';

Txn 2:
2a) y = SELECT cases FROM Country WHERE cname = 'Spain';
2b) y = y / 5;
2c) z = SELECT cases FROM Country WHERE cname = 'France';
2d) UPDATE Country SET cases = `y + z` WHERE cname = 'France';

9) Consider the following schedule. *Is it serializable?* ***If yes, write a serial order*** *that the transactions could execute in that would behave equivalently to this schedule.* ***If no,*** *then the schedule is incorrect at the Serializable isolation level;* ***write the strongest isolation level*** *at which this schedule would be correct.*

Assume that shared and exclusive locks can be used, and that locking granularity occurs at the row level. Reminder: the isolation levels, from weakest to strongest, are
None < Read Uncommitted < Read Committed < Repeatable Read < Serializable

2a) y = SELECT cases FROM Country WHERE cname = ‘Spain’;
1a) x = SELECT cases FROM Country WHERE cname = ‘France’;
2b) y = y / 5;
1b) x = x + 200;
1c) UPDATE Country SET cases = `x` WHERE cname = ‘France’;
2c) z = SELECT cases FROM Country WHERE cname = ‘France’;
2d) UPDATE Country SET cases = `y + z` WHERE cname = ‘France’;

Yes, this schedule is serializable. In fact, it is conflict serializable. It is equivalent to executing in the serial order [Txn1, Txn2]. It is also equivalent to [Txn2, Txn1] due to + associativity.

[7 points]
4 for stating serializability; 3 for stating a serial order
-1 if the student gave an isolation order

10) Consider the following schedule. *Is it serializable?* ***If yes, write a serial order*** *that the transactions could execute in that would behave equivalently to this schedule.* ***If no,*** *then the schedule is incorrect at the Serializable isolation level;* ***write the strongest isolation level*** *at which this schedule would be correct.*

Assume that shared and exclusive locks can be used, and that locking granularity occurs at the row level. Reminder: the isolation levels, from weakest to strongest, are
None < Read Uncommitted < Read Committed < Repeatable Read < Serializable

2a) y = SELECT cases FROM Country WHERE cname = 'Spain';
2b) y = y / 5;
2c) z = SELECT cases FROM Country WHERE cname = 'France';
1a) x = SELECT cases FROM Country WHERE cname = 'France';
1b) x = x + 200;
1c) UPDATE Country SET cases = `x` WHERE cname = 'France';
2d) UPDATE Country SET cases = `y + z` WHERE cname = 'France';

No, this schedule is not serializable. Read Committed is the strongest isolation level at which the schedule is correct.

It is not conflict serializable because there is a RW conflict between
2c) z = SELECT cases FROM Country WHERE cname = 'France';
and
1c) UPDATE Country SET cases = `x` WHERE cname = 'France';
and there is a WW conflict between
1c) UPDATE Country SET cases = `x` WHERE cname = 'France';
and
2d) UPDATE Country SET cases = `y + z` WHERE cname = 'France';

It is not serializable by manual inspection of the operations. Suppose `j` is the initial value of cases in France, and `c` is the initial value of cases in Spain. Executing the serial order [txn1, txn2] or [txn2, txn1] both result in the end value of cases in France of `j + 200 + c/5`.

With this schedule, the end value of cases in France is `j + c/5`. The `+200` is a lost update.

Reducing the isolation level to Repeatable Read will not help, because no inserts or deletes are present so the phantom problem is not present.

Reducing the isolation level to Read Committed helps, because it means that read locks no longer need to follow the 2-Phase-Locking protocol. A locking schedule could look like the following:
2) S(Spain)
2a) y = SELECT cases FROM Country WHERE cname = 'Spain';
2) U(Spain)

2b) y = y / 5;
2) S(France)
2c) z = SELECT cases FROM Country WHERE cname = ‘France’;
2) U(France)
1) S(France)
1a) x = SELECT cases FROM Country WHERE cname = ‘France’;
1) U(France)
1b) x = x + 200;
1) X(France)
1c) UPDATE Country SET cases = `x` WHERE cname = ‘France’;
1) Commit; U(France)
2) X(France)
2d) UPDATE Country SET cases = `y + z` WHERE cname = ‘France’;
2) Commit; U(France)

[8 points]
4 for stating not serializable; 4 for Read Committed
+2 If the student gave a schedule that’s consistent with their isolation level