CSE 344 Final Examination

Monday, March 18, 2019, 2:30-4:20

Name: _____

Question	Points	Score
1	35	
2	15	
3	10	
4	30	
5	20	
6	30	
7	60	
Total:	200	

- This exam is CLOSED book and CLOSED devices.
- You are allowed TWO, HAND-WRITTEN letter-size sheets with notes (both sides).
- You have 110 minutes;
- Answer the easy questions before you spend too much time on the more difficult ones.
- Good luck!

1 Relational Data Model

1. (35 points)

   A Web browser stores the local data in a relational database[1] with the following schema:

   ```
   History(url, ts)
   Cache(url, content, size)
   Bookmark(name, url)
   ```

   - Every time the user visits a page, the browsers inserts a record in `History`; `url` is the URL of the Webpage and has type `Text`; `ts` is the time stamp when the user accesses the page and has type `Int`.
   - Some Web pages are stored in the local cache of the browser; this is the table `Cache`; `content` is the HTML text of the page in the cache and has type `Text`, while `size` is its size in bytes and has type `Int`.
   - The user may create bookmarks and give them unique names; all bookmarks are stored in `Bookmark`.

   (a) (5 points) Sometimes users created duplicate bookmarks; they give two or more names to the same URL. Write a SQL query that returns all duplicate bookmarks. Your query should return a list of names and URLs, sorted by the URLs.

   Solution:

   ```
   -- drop/create tables for testing only
   drop table if exists history;
   drop table if exists bookmarks;
   drop table if exists cache;
   create table history(url text, ts int);
   create table cache(url text primary key, content text, size int);
   create table bookmark(name text primary key, url text);

   select distinct x.url, x.name
   from Bookmark x, Bookmark y
   where x.url = y.url and x.name != y.name
   order by x.url;
   ```

   -1 for missing distinct
   -1 for missing $x.name \neq y.name$
   $-1 \ldots -3$ for solutions that were too complex

   Several students turned in a wrong query (why?): there was no partial credit for a wrong answer.

[1] Chrome uses SQlite to store and manage all its data.

```
History(url, ts)
Cache(url, content, size)
Bookmark(name, url)
```

(b) (5 points) The browser wants to store a new page in the cache. There is no more space, and it decides to evict the oldest pages in order to make room. Write a SQL query that lists for each URL in the cache its size and the latest timestamp when that page was last accessed. Your query should return triples `url`, `size`, `ts`, ordered increasingly by `ts`.

Solution:

```
select y.url, max(x.ts) as tsmax, y.size
from History x, Cache y
where x.url = y.url
group by y.url, y.size
order by tsmax;
```

No credit for returning all time stamps ts (i.e. must have some max or something like this)
no penalty for nested subquery (this was lenient!)
no penalty for missing $y.size$ in the group by

Too many students got this problem wrong (why?)

```
History(url, ts)
Cache(url, content, size)
Bookmark(name, url)
```

(c) (15 points) The new page to be added to the cache has 1000 bytes, and the browser decides to evict from the cache the oldest pages in order to make room from the new page. For example, if there are four pages in the cache last accessed at time stamps 1,2,3,4 respectively, and their sizes are 500, 300, 300, 600, then the browser wants to delete the oldest three pages, since $500 + 300 + 300 \geq 1000$. Write a SQL query to return the URLs of all pages in the cache that the browser needs to delete to make room for 1000 bytes; your query should return a list of URL's (no need to actually delete them from `Cache`).

Note: only attempt to answer this question if you have answered question b. If you answered it, then you may refer to the query in b (no need to write it again).

Solution:

```
with tmp as
     (select y.url, max(x.ts) as tsmax, y.size
      from History x, Cache y
      where x.url = y.url
      group by y.url, y.size)
select x.url
from tmp x, tmp y
where x.url = y.url and y.tsmax < x.tsmax
group by x.url
having sum(y.size) < 1000;
```

This was a difficult question. I only gave partial credit for attempts that included a summation of $y.size$ for $y.tsmax < x.tsmax$ (or even for $y.tsmax > x.tsmax$, althought that more cumbersome to use). Depending on how this sum was used in the query, the partial credit ranged from 5 points to full points.

(d) In this question we will represent sparse arrays and matrices as relations. For example, this shows how a matrix A and a vector X might be represented:

The matrix $A = \begin{bmatrix} 0 & 0 & 5 \\ 2 & -7 & 0 \\ 0 & 0 & -1 \end{bmatrix}$ is represented as

A :

row	col	val
1	3	5
2	1	2
2	2	−7
3	3	−1

The array $X = \begin{bmatrix} 7 \\ 0 \\ -5 \\ 0 \end{bmatrix}$ is represented as

X :

pos	val
1	7
3	-5

Recall some standard definitions in linear algebra:

- Matrix/matrix product: $C = A \cdot B$, where $C_{ik} = \sum_j A_{ij} B_{jk}$.
- Matrix/vector product: $Y = A \cdot X$ where $Y_i = \sum_j A_{ij} X_j$.
- Vector/vector product: $X^t \cdot Y = \sum_i X_i Y_i$.
- The trace of a matrix is $\texttt{tr}(A) = \sum_i A_{ii}$.

A, B, C are matrices and X is a vector. For each of the SQL expressions below, write the corresponding formula in linear algebra.

For example, if the queries is:

```
select A.row, B.col, sum(A.val * B.val)
from A, B
where A.col = B.row
group by A.row,B.col;
```

then you answer $A \cdot B$.

If the query is:

```
select sum(A.val * B.val)
from A, B
where A.col = B.row and A.row = B.col;
```

then you answer $\texttt{tr}(A \cdot B)$, or $\texttt{tr}(B \cdot A)$ (both are correct answers).

Solution:

```
-- for testing only
drop table if exists A;
drop table if exists B;
drop table if exists C;
```

```
drop table if exists X;
create table A(row int, col int, val real);
create table B(row int, col int, val real);
create table C(row int, col int, val real);
create table X(pos int, val real);
```

i. (2 points)

```
select sum(A.val) from A where A.row = A.col;
```

i. $\texttt{tr}(A)$

Linear algebra expression:

ii. (2 points)

```
select sum(X1.val * A.val * X2.val)
from X X1, A, X X2
where X1.pos = A.col and A.row = X2.pos;
```

ii. $X^t \cdot A \cdot X$

Linear algebra expression:

iii. (2 points)

```
select sum(A.val) from A where A.row = A.col;
```

iii. $\texttt{tr}(A)$

Linear algebra expression:

iv. (2 points)

```
select A.row, A.col, A.val + sum(B.val*C.val)
from A, B, C
where A.row = B.row and B.col = C.row and C.col = A.col
group by A.row, A.col, A.val;
```

iv. $A + B \cdot C$

Linear algebra expression:

v. (2 points)

```
select sum(A.val*B.val*C.val)
from A, B, C
where A.col = B.row and B.col = C.row and C.col = A.row;
```

v. $\texttt{tr}(A \cdot B \cdot C)$

Linear algebra expression:

2 Datalog

2. (15 points)

In the kingdom of Datalandia there are nobles and commoners. When the kingdom was founded hundreds of years ago, the first king knighted some people who became the first noblemen, called *UrNobles*. The rule of the land is that a newborn becomes a noble if both his/her parents are nobles; otherwise he/she is a commoner. The kingdom meticulously maintains a database of all its subjects:

```
Person(pid,name); // every person in Datalandia who ever lived:
Father(fid,pid);  // fid is the father of pid
Mother(mid,pid);  // mid is the mother of pid
UrNoble(nid);     // the person ID's of the UrNobles
King(kid);        // the person ID's of all kings
```

(a) (10 points) Write a datalog query that returns the `pid`'s and names of all nobles.

Solution:

```
Noble(pid) :- UrNoble(pid)
Noble(pid) :- Nobel(fid),Nobel(mid),Father(fid,pid),Mother(fid,pid)
```

(b) (5 points) Unfortunately, something terribly wrong happened, and some of the kings of Datalandia were commoners. Write a datalog program to retrieve all the commoner kings of Datalandia. You should return a set of person IDs (`pid`) and names. You may use the datalog query written for the previous question.

Solution:

Solution: CommonKing(kid :- King(kid), not Noble(pid)

3 NoSQL, JSON, SQL++

3. (10 points)

   (a) (10 points) We are given a JSON file with noble prize laureates, with the following structure:

```
{"prizes": [
    { "year": "2018",
      "category": "physics",
      "overallMotivation": "For groundbreaking inventions in the field of laser physics",
      "laureates": [
        { "id": "960",
          "name": "Arthur Ashkin",
          "motivation": "\"for the optical tweezers and their application to biological systems\"",
          "share": "2"
        },
        { "id": "961",
          "name": "Grard Mourou",
          "motivation": "\"for their method of generating high-intensity, ultra-short optical pulses\"",
          "share": "4"
        },
        { "id": "962",
          "name": "Donna Strickland",
          "motivation": "\"for their method of generating high-intensity, ultra-short optical pulses\"",
          "share": "4"
        }
      ]
    },
    { "year": "2018",
      "category": "chemistry",
      ...
    },
    { "year": "2018",
      "category": "medicine",
      ...
    }
  ]
```

   Write a SQL++ query that returns each noble prize laureate who has received more than one award, along with a list of the years and categories that each such laureate has received. Your query should return a JSON file with a structure like the following:

```
{ "name": "Frederick Sanger",
  "awards": [ { "year": "1958", "category": "chemistry" },
              { "year": "1980", "category": "chemistry" } ]
}
{ "name": "Marie Curie, ne Sklodowska",
  "awards": [ { "year": "1903", "category": "physics" },
              { "year": "1911", "category": "chemistry" } ]
}
...
```

[this page is intentionally left blank]

Solution:

```
SELECT DISTINCT l.name, awards
FROM prizes p, p.laureates l
LET awards=(                          --Use a subquery to nest
  SELECT p2.year, p2.category         --DISTINCT is optional
  FROM prizes p2, p2.laureates l2
  WHERE l2.name = l.name
)
WHERE array_count(awards) > 1;        --coll_count() also works
```

```
Note: data adapted from: http://api.nobelprize.org/v1/prize.JSON

Using the real data, one solution is:

CREATE DATAVERSE noble;
CREATE TYPE noble.prizeType AS {auto_id:uuid};
CREATE DATASET noble.prize(noble.prizeType)
  PRIMARY KEY auto_id AUTOGENERATED;
LOAD DATASET noble.prize USING localfs
  (("path"="localhost://<path_to>/prize.json"),
   ("format"="json"));

WITH reduced AS (
  SELECT p.year, p.category,
         l.firstname || " " || l.surname AS name
  FROM noble.prize np, np.prizes p, p.laureates l
)
SELECT DISTINCT r1.name, awards
FROM reduced r1
LET awards=(
  SELECT year, category
  FROM reduced r2
  WHERE r2.name = r1.name
)
WHERE array_count(awards) > 1;
```

1 point for select clause, with some kind of collection
1 point for iterated from clause
1 point for distinct (or group by)
4 points for subquery
3 points for where / having clause

```
History(url, ts)
Cache(url, content, size)
Bookmark(name, url)
```

4 Query Execution and Optimization

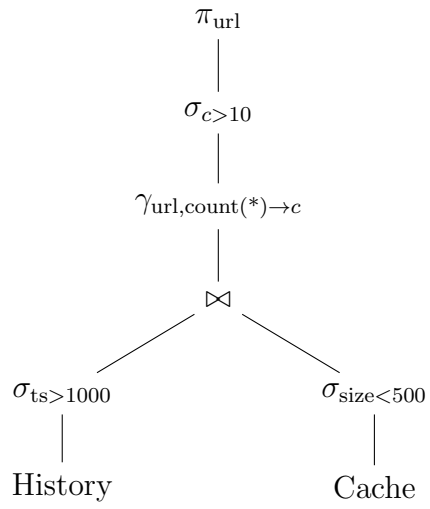
4. (30 points)

(a) (10 points) In this question we are using the browsing history schema again. Write a logical plan for the following query.

```
select x.url
from History x, Cache y
where x.url = y.url
  and x.ts > 1000 and y.size < 500
group by x.url
having count(*) > 10;
```

You should turn in a relational algebra tree.

Solution:



(b) In this question we consider three relations $R(A,B), S(B,C), T(C,D)$ and the following statistics:

$$\begin{aligned} T(R) &= 10^5 & B(R) &= 100 \\ T(S) &= 6 \cdot 10^6 & B(S) &= 3000 \\ T(T) &= 5 \cdot 10^4 & B(T) &= 40000 \\ V(R,A) &= 5 \cdot 10^4 \\ V(R,B) &= V(S,B) = 3 \cdot 10^3 \\ V(S,C) &= V(T,C) = 2 \cdot 10^4 \\ V(T,D) &= 10^4 \end{aligned}$$

i. (5 points) Estimate the number of tuples returned by $\sigma_{A=2432}(R)$. You should turn in an integer number.

Solution:

$$\frac{T(R)}{V(R,A)} = \frac{10^5}{5 \cdot 10^4} = 2$$

ii. (5 points) Estimate number of tuples returned by the following query:

```
SELECT *
FROM R, S, T
WHERE R.A = 2432 and R.B = S.B and S.C = T.C and T.D = 1234
```

You should turn in an integer number.

Solution:

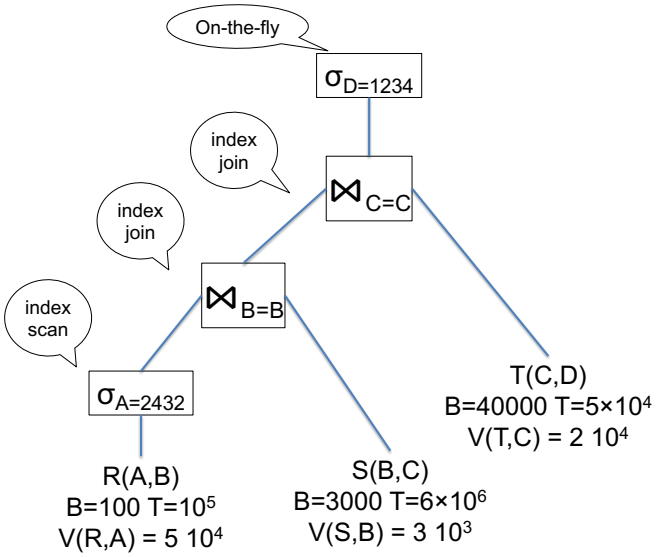
$$\frac{T(R)T(S)T(T)}{V(R,A)V(R,B)V(T,C)V(T,D)} = \frac{10^5 \cdot 6 \cdot 10^6 \cdot 5 \cdot 10^4}{5 \cdot 10^4 \cdot 3 \cdot 10^3 \cdot 2 \cdot 10^4 \cdot 10^4} = 1$$

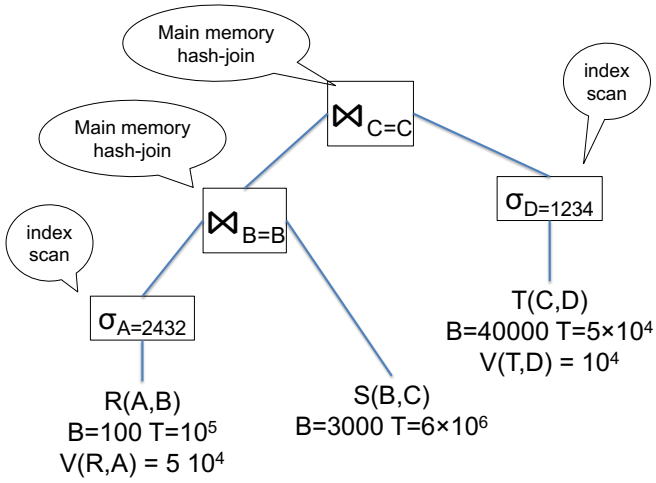
Here and in the previous question, significant partial credit was given if work shown (usually identifying correct variables) was close to what was correct.

iii. (10 points) Assume the following indices:

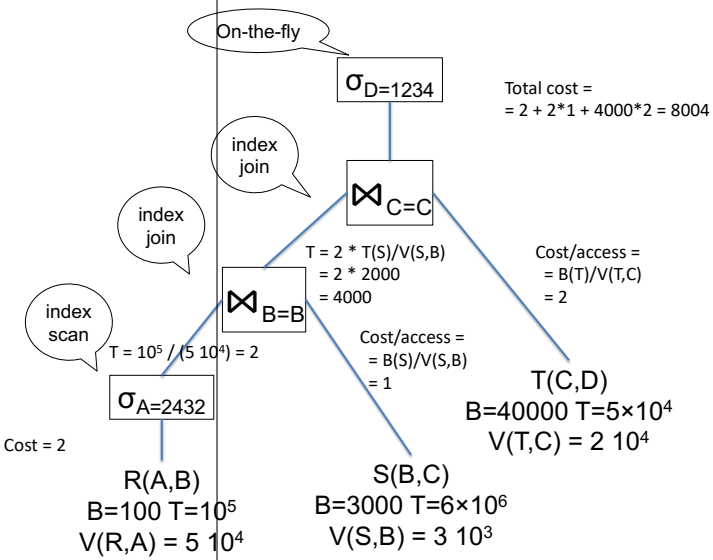
- Unclustered indexes on $R.A$ and $R.B$
- Clustered index in $S.B$, unclustered index on $S.C$.
- Clustered indexe on $T.C$, unclustered index on $T.D$.

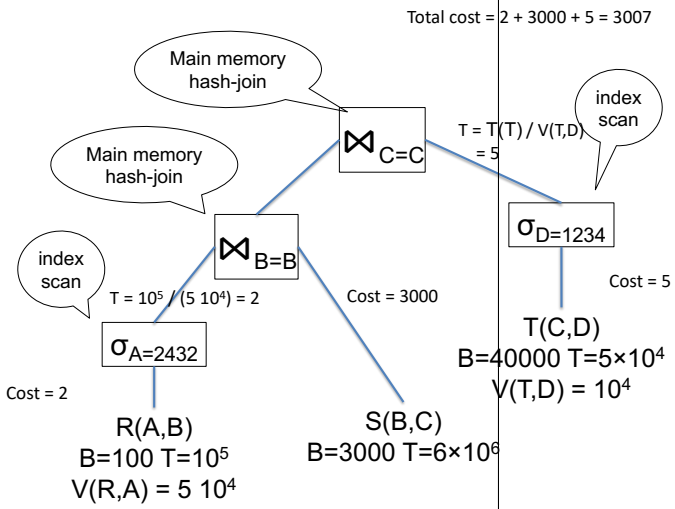
Estimate the I/O cost for two the physical plans below. Use the same statistics as in the previous question (they are shown on the plans, for your convenience).





Solution:





For each plan:
about 1/5 if they found one sub answer
about 3/5 if they found two sub answers
Points varied based on amount of work shown.

5 Parallel Query Processing

5. (20 points)

Consider two relations with the following schema and statistics:

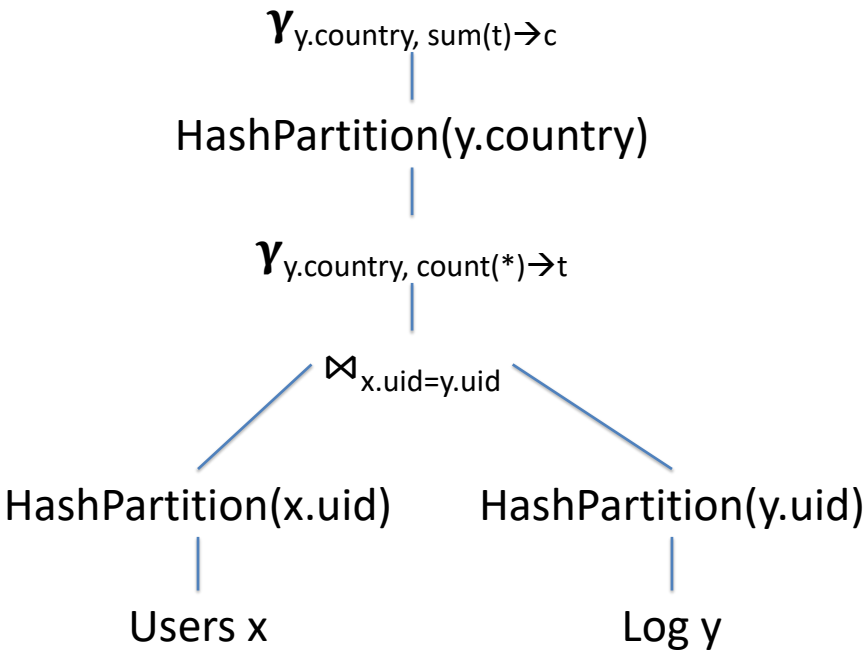
$$\texttt{Users(}\underline{\texttt{uid}}\texttt{, name, country)}$$
$$\texttt{Log(uid, url)}$$
$$T(\texttt{Users}) = 10,000,000$$
$$T(\texttt{Log}) = 20,000,000,000$$
$$V(\texttt{Users}, \texttt{country}) = 100$$
$$V(\texttt{Log}, \texttt{uid}) = 5,000,000$$

The data is initially block partitioned on 1000 servers, so that each server holds 10000 records; we assume that, initially, the records of both `Users` and `Log` are randomly distributed to the 1000 servers,

The query below counts the number log entries from each country:

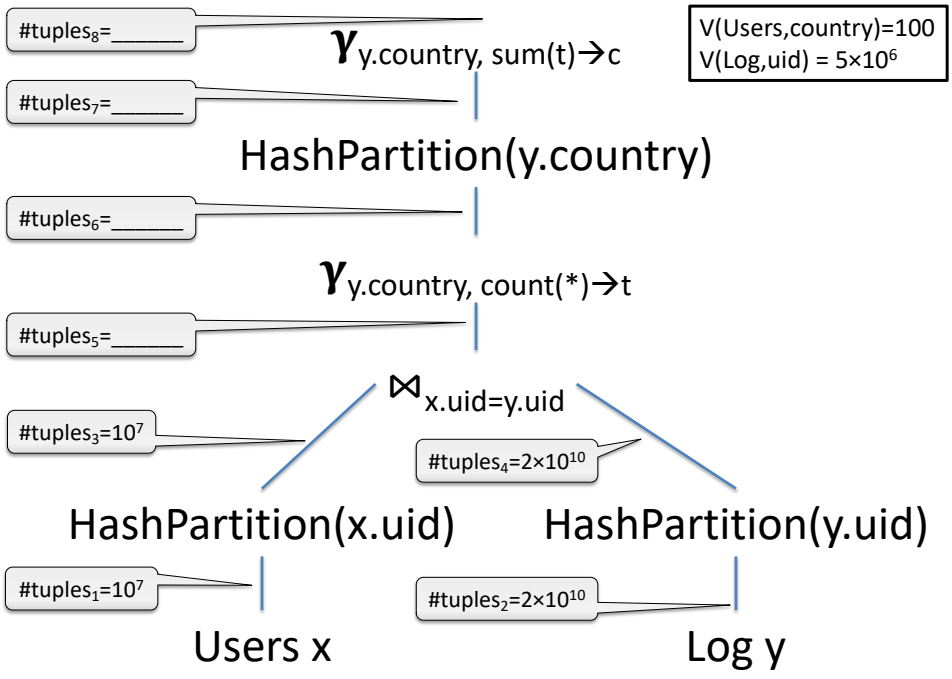
```
select x.country, count(*) as c
from Users x, Log y
where x.uid = y.uid
group by x.country
```

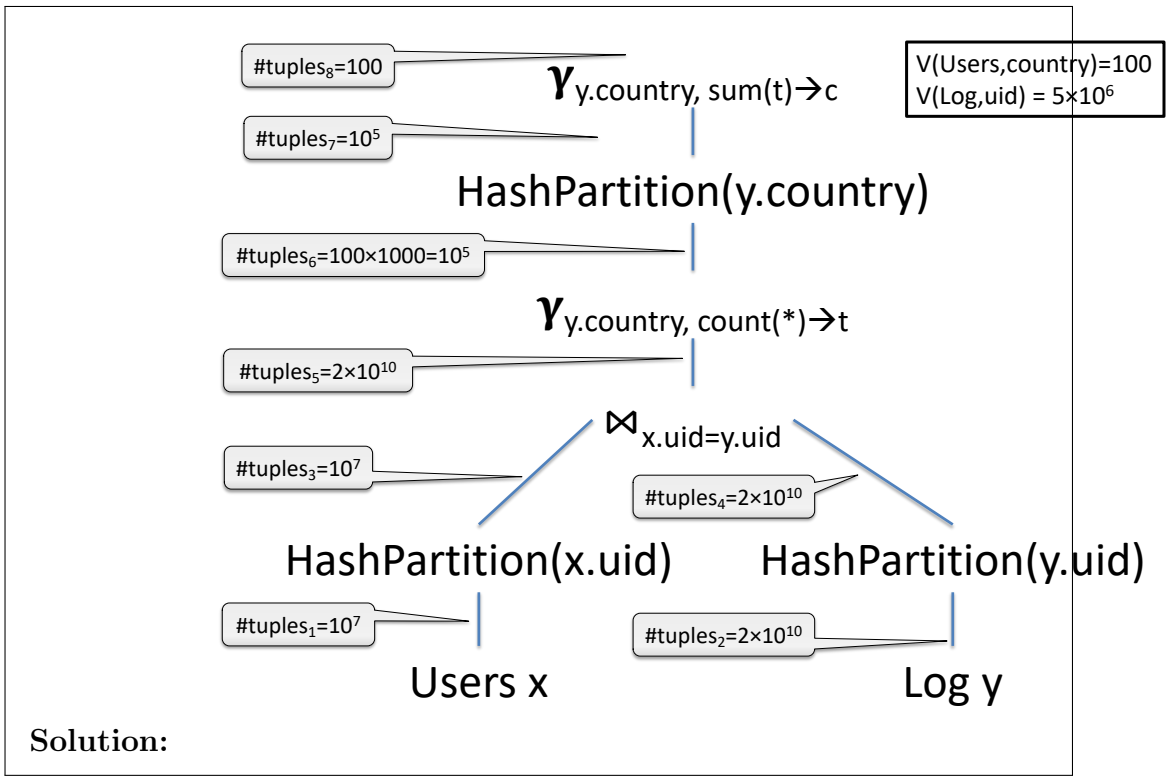
(a) The query optimizer chooses to compute the query using the following plan:



In words, the data is first hash-partitioned on `uid`, followed by a local join and a local group-by, followed by a repartition on `country`, and a final group-by.

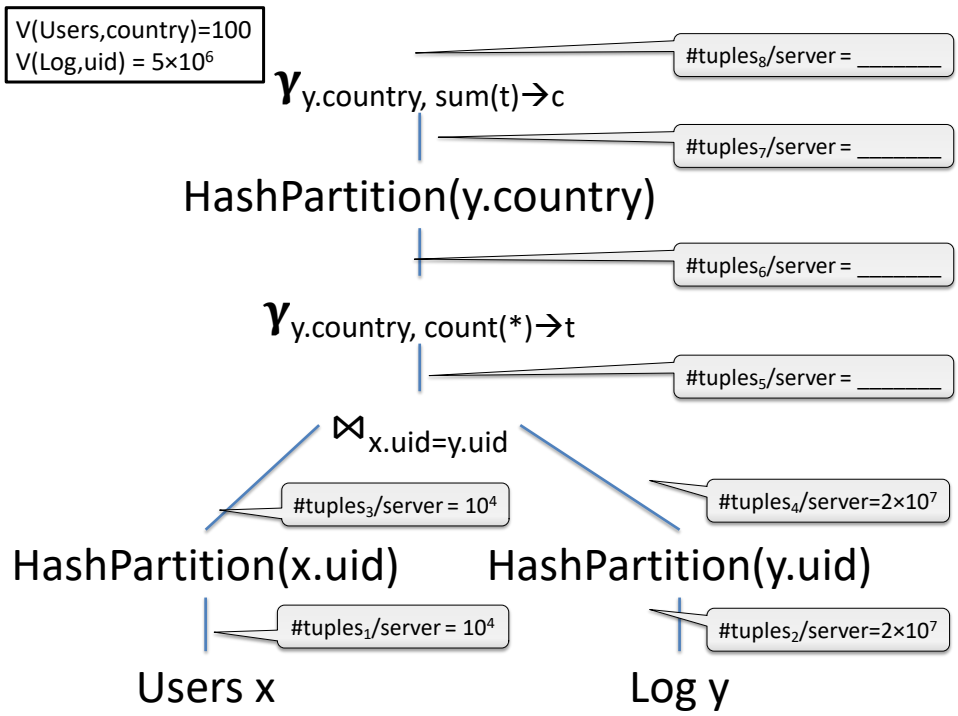
i. (4 points) Estimate the size (number of tuples) of each intermediate relation in the plan. You may assume that the data is uniformly distributed and that the attributes are independent. Write your answer in the figure below, by filling out each missing `#tuples`. Notice that `#tuples` represents the total number of tuples, from all servers.

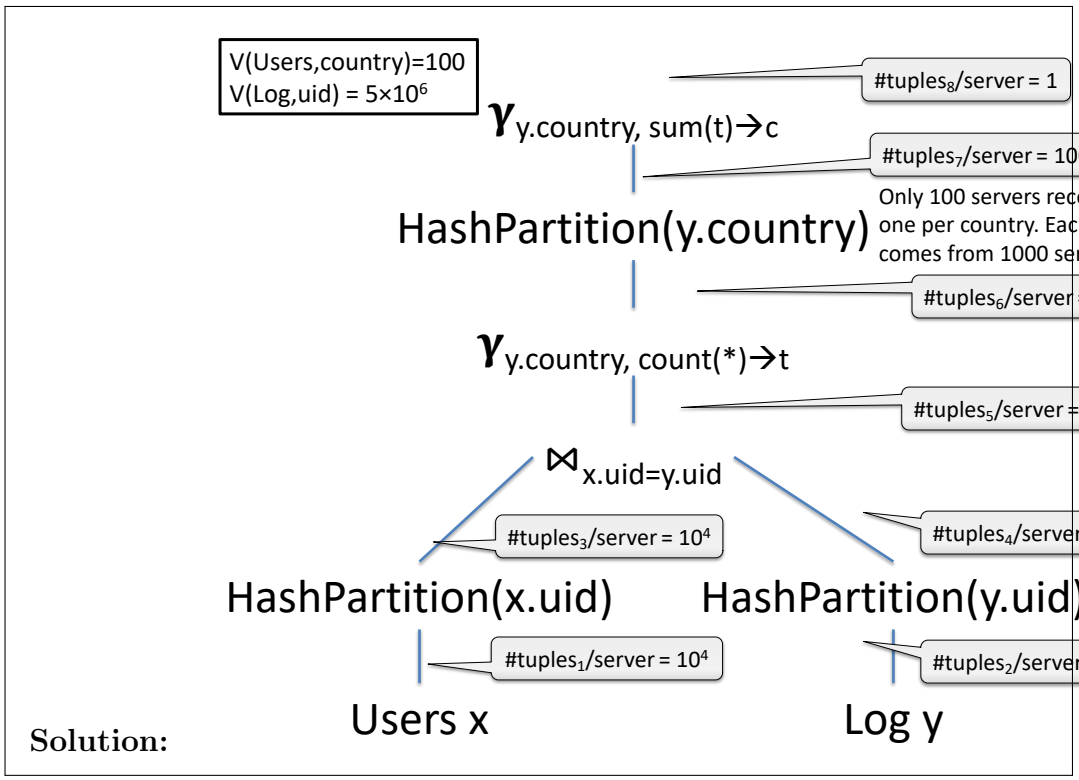



#tuples8=100
γy.country, sum(t)→c
V(Users,country)=100
V(Log,uid) = 5×10^6
#tuples7=10^5
HashPartition(y.country)
#tuples6=100×1000=10^5
γy.country, count(*)→t
#tuples5=2×10^10
⋈x.uid=y.uid
#tuples3=10^7
#tuples4=2×10^10
HashPartition(x.uid)
HashPartition(y.uid)
#tuples1=10^7
#tuples2=2×10^10
Users x
Log y

Solution:

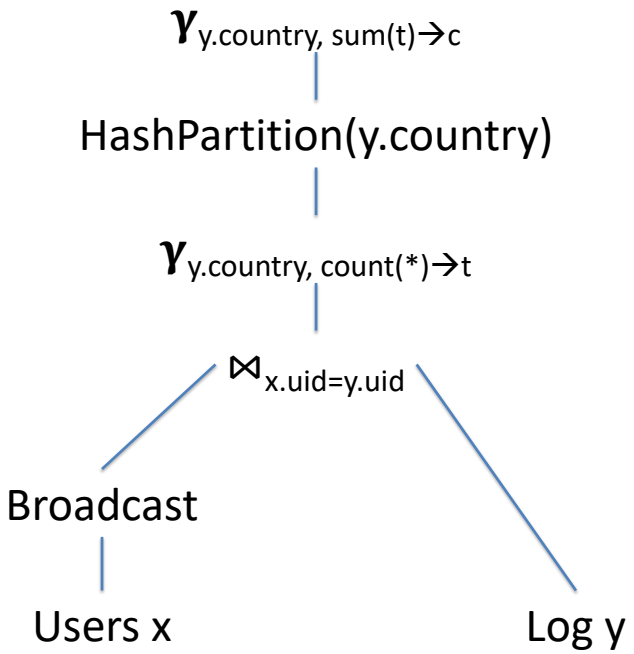
ii. (3 points) Assuming the system uses 1000 servers to compute the query, indicate the number of tuples per server at each step (i.e. the load per server). Assume the data is uniformly distributed, in the best possible way. In the case when not all servers receive the same number of tuples, then indicate the largest number. (This question should be easy to answer if you answered the previous one.)





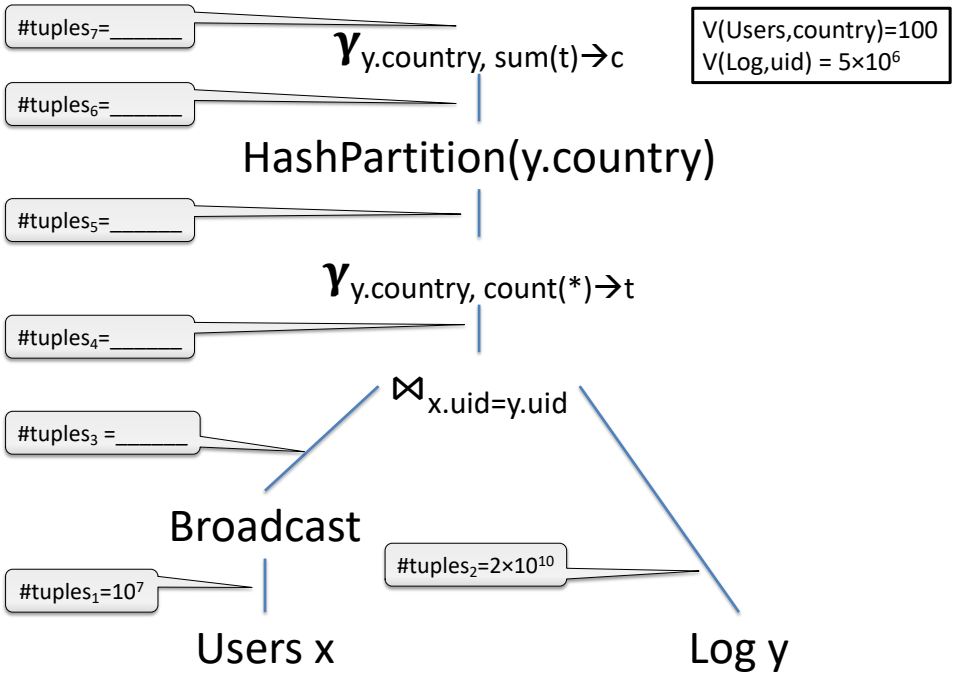
iii. (3 points) Now assume that the data is not uniform. What is the largest possible number of tuples received by any server, and which intermediate result creates this largest number of tuples? Write your answers by referring to the figure above, for example you may write $\texttt{tuples}_4/\texttt{server} = 7 \cdot 10^{19}$ (not a real answer).

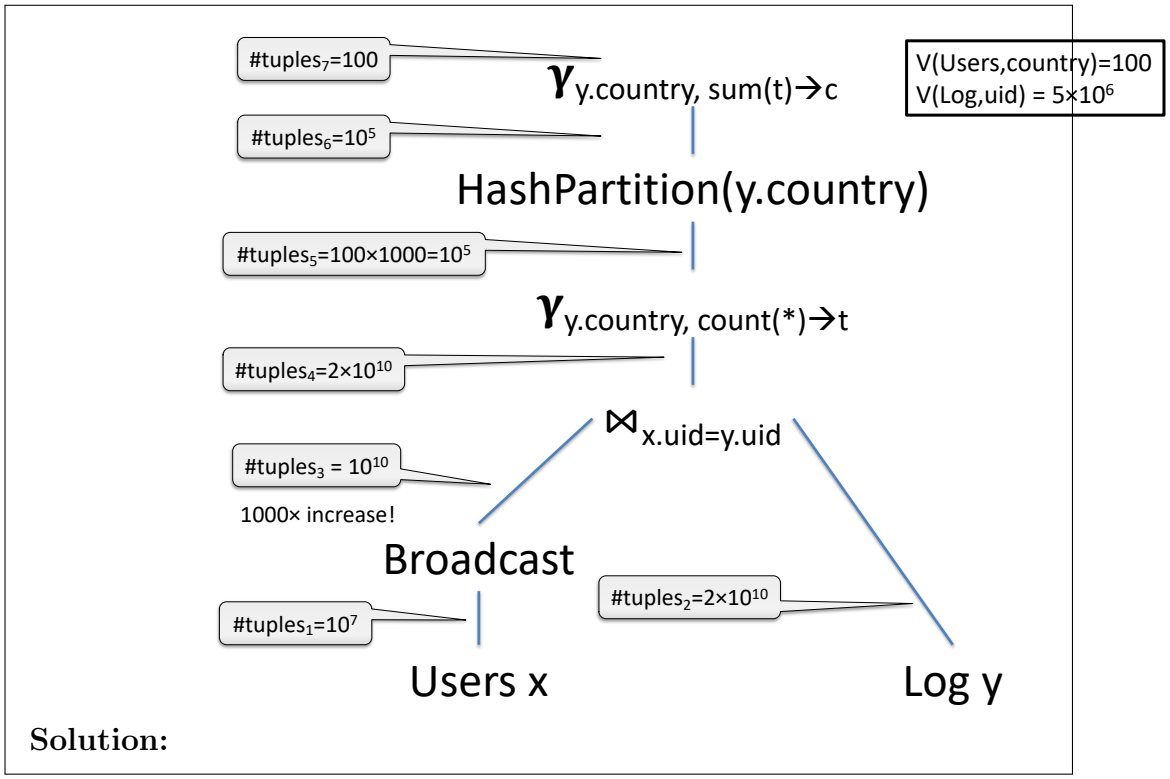
(b) Now the query optimizer chooses to compute the query using the following plan:



In other words, it first broadcasts `Users`, then computes the local join followed by a local group-by, then reshuffles the data based on `country`, followed by a final local group-by. Answer the same questions as before:

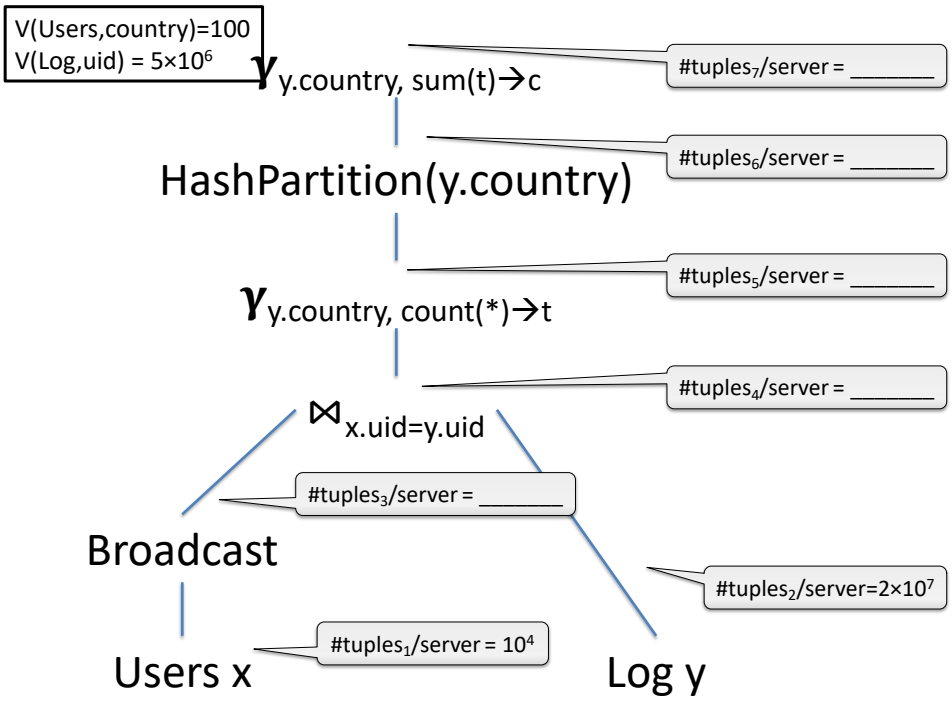
i. (4 points) Estimate the size (number of tuples) of each intermediate relation in the plan, as before.

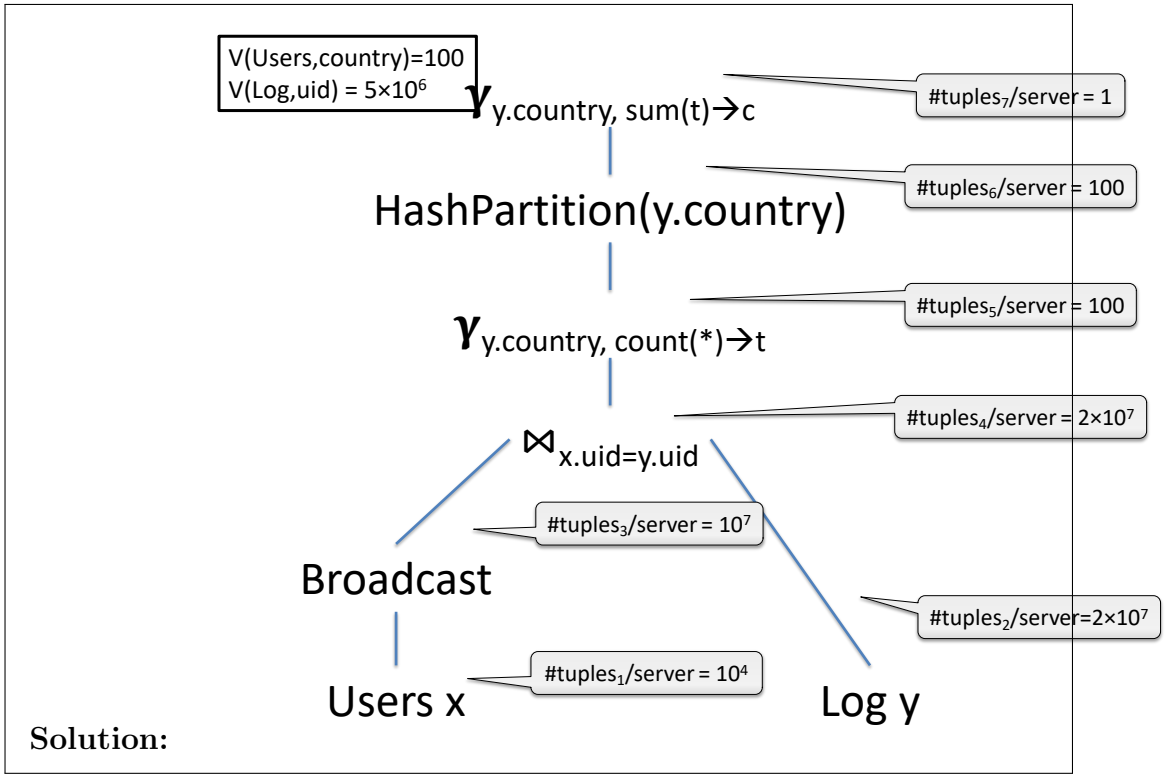




Solution:

ii. (3 points) Assuming the system uses 1000 servers to compute the query, indicate the number of tuples per server at each step (i.e. the load per server), as before.





Solution:

iii. (3 points) Now assume that the data is not uniform. What is the largest possible number of tuples received by any server, and which intermediate result creates this largest number of tuples?

6 Conceptual Design

6. (30 points)

(a) (10 points) Consider a relation $R(A, B, C, D, E)$ satisfying the following FD's:

$$AB \rightarrow CD$$
$$DE \rightarrow B$$

Decompose R into BCNF.

Solution:

- $\{DE\}^+ = BDE$. Decompose into $R_1(\underline{D, E}, B)$, $R_2(A, C, D, E)$.

Alternative:

- $\{AB\}^+ = ABCD$. Decompose into $R_1(\underline{A, B}, C, D)$, $R_2(A, B, E)$.

(b) (10 points) Consider two relations $R(\underline{K}, A, B)$, $S(\underline{L}, C, D)$. The following query returns a relation with attributes K, A, B, L, C, D:

```
select *
from R, S
where R.B=S.L and R.A=S.D
```

Find all functional dependencies satisfied by the answer to the query above. You only need to indicate a minimal set of FDs; for example if you wrote $X \to Y$ and $YZ \to U$ then you don't need to write $XZ \to U$.

Solution:

$K \to LABCD$ $\quad$ $L \to BCD$ $\quad$ $B \to L$

$A \to D$ $\quad$ $D \to A$

(c) Consider a relation with three attributes A, B, C. Answer the questions below.

i. (2 points) Give an example of functional dependencies such that the relation has a single key consisting of two attributes.

Solution: $AB \to C$

ii. (2 points) Give an example of functional dependencies such that the relation has two keys consisting of one attribute each.

Solution: $A \to BC$, $B \to AC$.

iii. (2 points) Give an example of functional dependencies such that both AB and AC are keys, but A is not a key.

Solution: $AB \to C$, $AC \to B$.

iv. (2 points) Give an example of functional dependencies such that the relation is *not* in BCNF.

Solution: $A \to B$.

v. (2 points) Give an example of functional dependencies such that the closures A^+, B^+, C^+ are three distinct sets, meaning no two sets can be equal.

Solution: $A \to B$, $B \to C$. Then $A^+ = ABC$, $B^+ = BC$, $C^+ = C$.

7 Transactions

7. (60 points)

(a) For each schedule below indicate whether it is conflict serializable and, if it is, indicate the equivalent serial schedule.

i. (5 points)

$$W_1(B), R_3(A), W_2(A), R_2(C), R_3(B), R_1(C), W_4(C)$$

Solution: Conflict serializable, in the unique order $1, 3, 2, 4$
TO DRAW THE GRAPH.

ii. (5 points)

$$W_1(B), R_3(A), W_2(A), R_2(C), R_3(B), W_4(C), R_1(C)$$

Solution: Not conflict serializable.
TO DRAW THE GRAPH

(b) A concurrency manager uses strict 2PL. The system runs only three transactions concurrently, denoted T_1, T_2, T_3. In each case below indicate whether there exists a schedule that leads to a deadlock. If you answer yes, then write a schedule that leads to a deadlock (up to the deadlock).

i. (5 points) The transactions are:

$$T_1 : R_1(A), W_1(B)$$
$$T_2 : R_2(B), W_2(C)$$
$$T_3 : R_3(C), W_3(A)$$

i. **Yes**

Can this lead to a deadlock?

Solution:

$$R_1(A), R_2(B), R_3(C), \underbrace{W_1(B)}_{\texttt{deadlock}}$$

ii. (5 points) The transactions are:

$$T_1 : R_1(A), W_1(B), W_1(C)$$
$$T_2 : W_2(A), R_2(B), W_2(C)$$
$$T_3 : W_3(A), W_3(B), R_3(C)$$

ii. **No**

Can this lead to a deadlock?

iii. (5 points) The transactions are:

$$T_1 : W_1(D), R_1(A), W_1(B)$$
$$T_2 : W_2(D), R_2(B), W_2(C)$$
$$T_3 : W_3(D), R_3(C), W_3(A)$$

iii. **No**

Can this lead to a deadlock?

(c) (5 points) Consider the following three transactions, where St_i indicates the start of T_i and Co_i indicates the commit of T_i:

$$T_1 : St_1, R_1(A), W_1(B), Co_1$$
$$T_2 : St_2, R_2(B), W_2(C), Co_2$$
$$T_3 : St_3, R_3(C), W_3(A), Co_3$$

Give an example of a schedule with the following properties: (1) the schedule is serializable; (2) transaction T_1 ends before transaction T_3 begins; (3) the only serialization order is T_3, T_2, T_1. In your schedule include the St_i and Co_i actions.

Solution: This problem was wrong! We only discovered during grading. Every student received full credit (5 points).

The correct question should be without a conflict on A, e.g.:

$$T_1 : St_1, R_1(A), W_1(B), Co_1$$
$$T_2 : St_2, R_2(B), W_2(C), Co_2$$
$$T_3 : St_3, R_3(C), Co_3$$

Then the solution is:

$$St_1, R_1(A), St_2, R_2(B), W_1(B), Co_1, St_3, R_3(C), W_2(C), Co_2, Co_3$$

(d) (10 points) The SQL standard defines three *weak isolation levels*: dirty reads, read committed, and repeatable reads. As you know:

- *Dirty reads* means no read locks.
- *Read committed* means short-duration read locks.
- *Repeatable reads* means long-duration read locks (full 2PL).

Consider two transactions T_1, T_2. For each schedule below, indicate under which isolation level that schedule is possible:

Time $\rightarrow$

Schedule$_1$:	St_1,	St_2,	$R_1(A)$,	$W_2(A)$,	$W_2(B)$,	$R_1(B)$,	Co_2	$W_1(A)$, Co_1
Schedule$_2$:	St_1,	St_2,	$R_1(A)$,	$W_2(A)$,	$W_2(B)$,	Co_2	$R_1(B)$,	$W_1(A)$, Co_1
Schedule$_3$:	St_1,	St_2,	$W_2(A)$,	$R_1(A)$,	$R_1(B)$,	$W_2(B)$,	Co_2	$W_1(A)$, Co_1
Schedule$_4$:	St_1,	St_2,	$W_2(A)$,	$R_1(A)$,	$R_1(B)$,	$W_1(A)$,	Co_1	$W_2(B)$, Co_2

Write yes/no answers below:

	Dirty reads	Read committed	Repeatable Reads
Schedule$_1$			
Schedule$_2$			
Schedule$_3$			
Schedule$_4$			

Solution:

	Dirty reads	Read committed	Repeatable Reads
$\texttt{Schedule}_1$	yes	no	no
$\texttt{Schedule}_2$	yes	yes	no
$\texttt{Schedule}_3$	yes	no	no
$\texttt{Schedule}_4$	no	no	no

First two errors: no penalty. The remaining errors: one point penalty for each.

(e) For each of the following statements indicate whether it is true or false:

i. (2 points) If there are an odd number of transactions, then deadlock can never occur.

i. **False**

True or false?

ii. (2 points) In a static database, every serializable schedule is conflict serializable.

ii. **False**

True or false?

iii. (2 points) In a dynamic database, every serializable schedule is conflict serializable.

iii. **False**

True or false?

iv. (2 points) In a static database, every conflict serializable schedule is serializable.

iv. **True**

True or false?

v. (2 points) In a dynamic database, every conflict serializable schedule is serializable.

v. **False**

True or false?

vi. (2 points) T_1 holds a shared lock on A. When T_2 requests a shared lock on A, the scheduler will grant it

vi. **True**

True or false?

vii. (2 points) T_1 holds a shared lock on A. When T_2 requests an exclusive lock on A, the scheduler will grant it

vii. **False**

True or false?

viii. (2 points) A concurrency management system uses strict 2PL, with shared locks for reads and and exclusive locks for writes. If all transactions are read-only, then deadlock is not possible.

viii. **True**

True or false?

ix. (2 points) An OLAP workload ("Online analytical processing") means a workload consisting of simple queries and many updates.

ix. **False**

True or false?

x. (2 points) An OLTP workload ("Online transaction processing") means a workload consisting of simple queries and may updates.

x. **True**

True or false?