CSE 344 Final Examination

Monday, December 10, 2018, 2:30-4:20

Name: ____________________________

Question	Points	Score
1	30	
2	20	
3	20	
4	30	
5	20	
6	35	
7	45	
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. (30 points)

You are now a Climate Scientist, and run simulations to predict future changes in the climate. Your job is to improve the simulator in order to improve the climate predictions. Every time you have a new version of the simulator, you run it, and it will generate a dataset where, for each year, it records the predicted temperature at each location of your simulated area. You analyze the results, improve the simulator, and repeat. As a good scientist, you keep the results of all simulations in a relation with the following schema:

`Sim(version, year, loc, temp)`

The data types of `Version` and `year` are integers; `temp` is a real number; and `loc` is a text.

(a) (5 points) Write a SQL CREATE TABLE statement to create the `Sim` relation. Identify the key in Sim, and create the key in your SQL statement. Choose `float` and for the real number type and `text` for the text type.

Solution:

```
drop table if exists sim;

create table sim(
   version int,
   year int,
   loc text,
   temp float,
   constraint pk primary key (version, year, loc)
);
```

1 point off for wrong or missing key

`Sim(version, year, loc, temp)`

(b) (5 points) Simulation version 13 is wrong, due to a bug in the simulator. Write a SQL query to delete version 13 from your dataset.

Solution:

```
delete from sim where version = 13;
```

Almost everyone got this right.

`Sim(version, year, loc, temp)`

(c) (10 points) Write a SQL query that returns all locations where the temperature has increased in every year of the simulation with version number 23.

Solution:

```
select distinct x.loc
from sim x
where x.version = 23 and
  not exists (select * from sim y, sim z
              where y.version = 23 and z.version = 23
              and x.loc = y.loc and x.loc = z.loc
              and y.year < z.year and y.temp > z.temp);
```

2 points off for `x.verion = 23` inside the subquery
8 points off for no subquery
4 points off for single `sim y` in the subquery (doesn't work)

`Sim(version, year, loc, temp)`

(d) (10 points) Now you are comparing versions 22 and 23. Write a SQL query that lists all years where the maximum temperature (at all locations) in version 22 was lower than the minimum temperature in the same year in version 23.

Solution:

```
select distinct x.year
from sim x, sim y
where x.version = 22 and y.version = 23
  and x.year = y.year
group by x.year
having max(x.temp) < min(y.temp);
```

1 point off for missing `x.year=y.year`
2-3 points off for missing group-by
various points off for various outher mistakes

2 Datalog

2. (20 points)

   A social network site has a database with its users and blogs that they post:

   ```
   User(uid, name, email)
   Blog(bid, author, text)
   Follows(uid1, uid2)
   ```

   Here `Blog.author`, `Follows.uid1` and `Follows.uid2` are foreign keys to `User`. A record in `Follows` means that the user with `uid1` follows the user with `uid2`.

   On this social netwok a user can only see her own blogs, and all the blogs seen by all the users she follows. For example if Alice follows Bob and Bob follows Carol, then Alice can read her blogs, Bob's blogs, and Carol's blogs, while Bob can read his blogs and Carol's blogs.

   (a) (10 points) The blog with `bid` = 359 contains misleading information. The site administrator wants to contact all users who can see this blog. Write a datalog program to find all users who the site owner needs to contact. Your program should return a set of `uid, name, email` tuples.

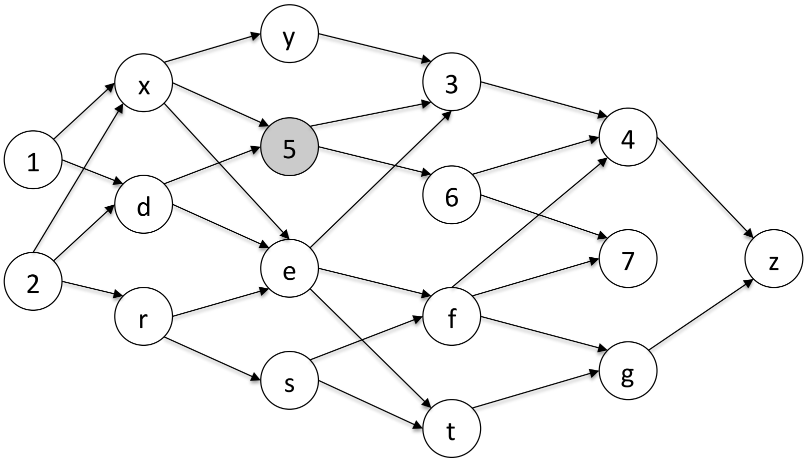
Solution:

```
q(uid) :- Blog(359,uid, -)
q(p1) :- q(p2), Follows(p1, p2)
answer(p, name, email) :- q(p), User(p,name,email)
```

(b) (10 points) Consider the datalog program below:

```
T(x,y) :- R(x,z),R(y,z)
T(x,y) :- T(x,z),T(z,y)
Q(w)   :- T(5,w)
```

Show the output of the IDB predicate Q when the program is run on the relation R given by the graph below; for example, the edge $2 \to d$ denotes the tuple $(2, d) \in R$. You do not need to justify your answer.



Solution: $5, y, e, s$

3 NoSQL, JSON, SQL++

3. (20 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;
```

My grading rubric:
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

(b) For each statement below indicate if it is true or false.

i. (2 points) JSON is in First Normal Form.

i. **false**

True or false?

ii. (2 points) JSON represents semistructured data.

ii. **true**

True or false?

iii. (2 points) It is easy to represent a many-to-one relationship in JSON, but it is difficult to represent a many-to-many relationship.

iii. **true**

True or false?

iv. (2 points) SQL++ can express all queries expressible in datalog.

iv. **false**

True or false?

v. (2 points) ACID transactions today are increasingly using JSON data as opposed to relational data.

v. **false**

True or false?

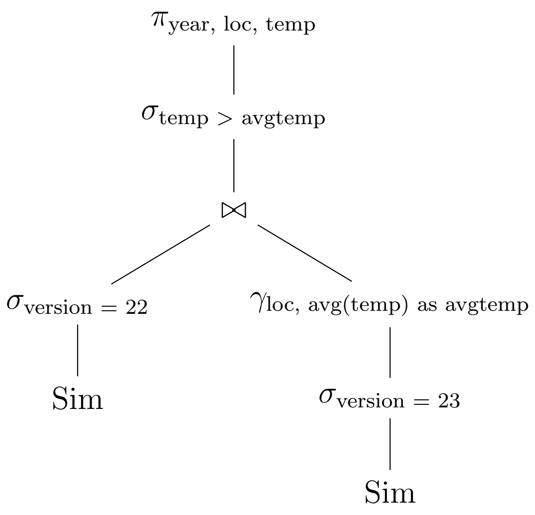
4 Query Execution and Optimization

4. (30 points)

   (a) (10 points) Write a logical plan for the following query.

```
select x.year, x.loc, x.temp
from Sim x
where x.version = 22
 and x.temp > (select avg(y.temp)
               from sim y
               where y.version = 23
                 and x.loc = y.loc);
```

You should turn in a relational algebra tree.


$\pi_{\text{year, loc, temp}}$
$\sigma_{\text{temp > avgtemp}}$
$\bowtie$
$\sigma_{\text{version = 22}}$
$\gamma_{\text{loc, avg(temp) as avgtemp}}$
Sim
$\sigma_{\text{version = 23}}$
Sim

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.

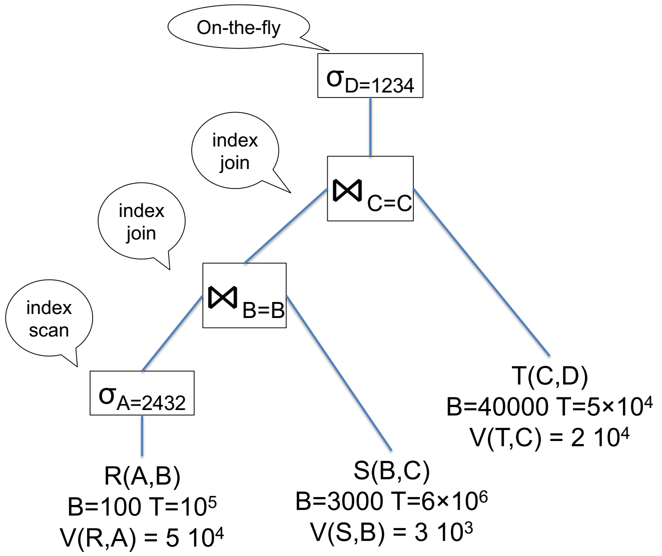
Solution:

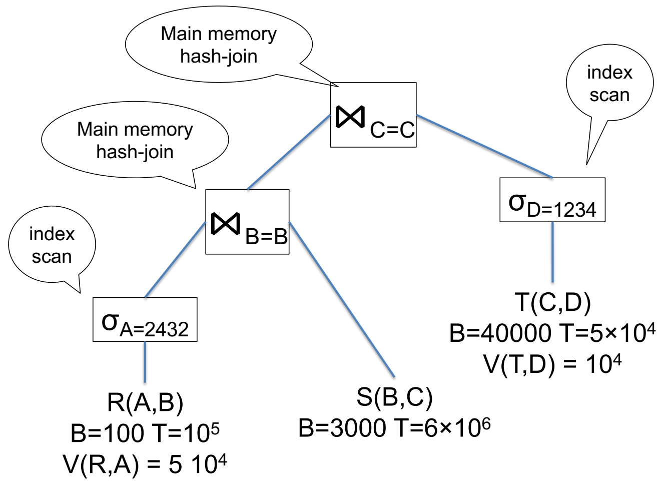
$$\frac{T(R)T(S)T(T)}{V(R, A)V(R, B)V(T, C)V(T, D)} = \frac{10^5 \cdot 6 \cdot 10^6 \cdot 5 \cdot 10^4}{5 \cdot 10^4 \cdot 3 \cdot 10^3 \cdot 2 \cdot 10^4 \cdot 10^4} = 1$$

iii. (10 points) Assume the following indices:

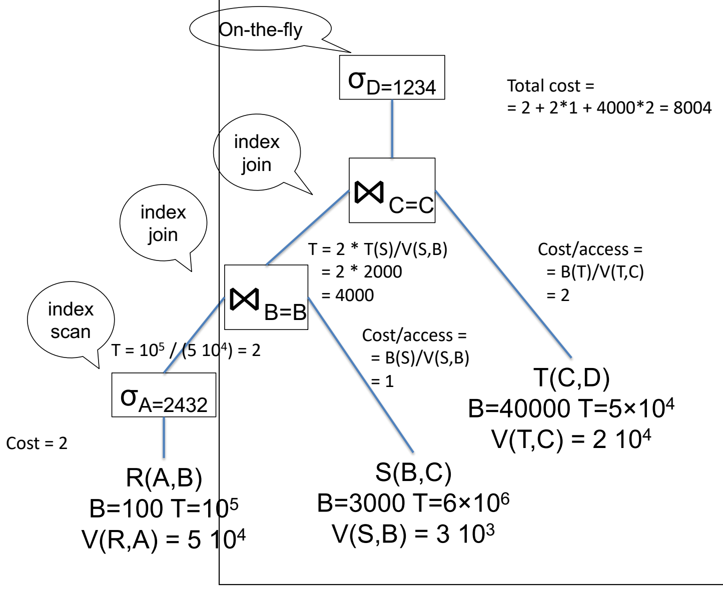
- Unclustered indexes on $R.A$ and $R.B$
- Clustered index in $S.B$, unclustered index on $S.C$.
- Clustered indexe on $T.C$, unclustered index on $T.D$.

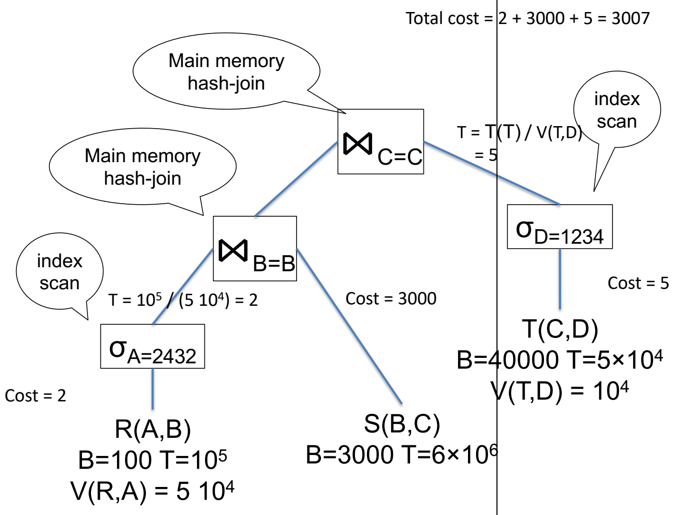
Estimate the I/O cost for two the physical plans below. Use the same statistics as in the previous question (they are shown on the plans, for your convenience).





Solution:





5 Parallel Query Processing

5. (20 points)

Consider a relation with the following schema and statistics:

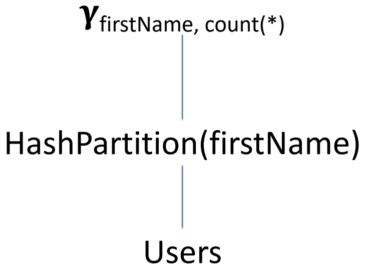
$$\texttt{Users(}\underline{\texttt{uid}}\texttt{,firstName,lastName)}$$
$$T(\texttt{Users}) = 10,000,000$$
$$V(\texttt{Users}, \texttt{firstName}) = 100,000$$

The query below counts the number of users that have the same first name:

```
select firstName, count(*)
from Users
group by firstName
```

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

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



In words, the data is first reshuffled among the 1000 servers using by hash partitioning on `firstName`, then each server computes a group-by locally.

Answer the questions below:

i. (2 points) The most popular first name is Mary. Suppose there are 410 users called Mary. In expectation, what is the maximum number of records received by any server during reshuffling? Choose the closest:

(a) 1

(b) 100

(c) 410

(d) 10000

(e) 410000

(f) 10M

i. **(d)**

Choose from (a)-(f)

ii. (2 points) The most popular first name is Mary. Suppose there are 140000 users called Mary. In expectation, what is the maximum number of records received by any server? Choose the closest:

(a) 1

(b) 100

(c) 140

(d) 10000

(e) 140000

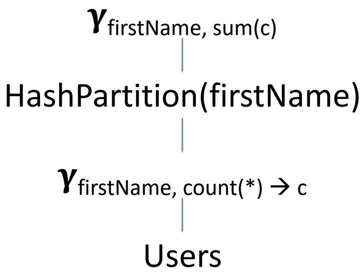
(f) 10M

ii. **(e)**

Choose from (a)-(f)

`Users(uid,firstName,lastName)`

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



$\gamma_{\text{firstName, sum(c)}}$
HashPartition(firstName)
$\gamma_{\text{firstName, count(*)} \rightarrow \text{c}}$
Users

Answer the questions below:

i. (2 points) The most popular first name (Mary) occurs 410 times. In expectation, what is the maximum number of records received by any server? Choose the closest:

(a) 1
(b) 100
(c) 410
(d) 10000
(e) 410000
(f) 10M

i. **(b)**

Choose from (a)-(f)

ii. (2 points) The most popular first name (Mary) occurs 140000 times. In expectation, what is the maximum number of records received by any server? Choose the closest:

(a) 1
(b) 100
(c) 140
(d) 10000
(e) 140000
(f) 10M

ii. **(b)**

Choose from (a)-(f)

(c) For each statement below indicate if it is true or false.

i. (2 points) For any query Q, if you can compute Q on a single server in one hour, then you can compute it in parallel on 60 servers in 1 minute.

i. **False**

True or false?

ii. (2 points) For any query Q, if you can compute Q in parallel on 60 servers in 1 minute, then you can compute it on a single server in one hour.

ii. **True**

True or false?

iii. (2 points) Spark improved over MapReduce by introducing the idea of distributing the data over many servers.

iii. **False**

True or false?

iv. (2 points) Spark improved over MapReduce by introducing the idea of allowing recovery from failure without having to store all interemediate results on disk.

iv. **True**

True or false?

v. (2 points) Consider the function in Spark $\texttt{map}(f) : \texttt{RDD} < T > \rightarrow \texttt{RDD} < U >$, where $f : T \rightarrow U$. The output RDD can never have a strictly smaller cardinality (number of tuples) than that of the input RDD.

v. **True**

True or false?

vi. (2 points) Consider the function $\texttt{flatMap}(f) : \texttt{RDD} < T > \rightarrow \texttt{RDD} < U >$, where $f : T \rightarrow \texttt{Seq}(U)$. The output RDD can never have a strictly smaller cardinality than that of the input RDD. (Recall: $\texttt{Seq}(T)$ means *sequence of* T, or *array of* T.)

vi. **False**

True or false?

6 Conceptual Design

6. (35 points)

(a) (5 points) Consider the following table:

A	B	C	D	E
0	0	0	0	0
1	1	1	1	1
2	2	2	2	0
3	3	3	0	1
4	4	0	1	0
5	5	1	2	1
6	0	2	0	0
7	1	3	1	1
8	2	0	2	0
9	3	1	0	1
10	4	2	1	0
11	5	3	2	1
12	0	0	0	0

Find all functional dependencies that hold on this table. You only need to write a minimal set of functional dependencies that logically imply all others. Hint: notice that $B = A \mod 6$ etc; you should be able to find the FD's very fast.

Solution: $A \to BCDE$, $C \to E$, $B \to DE$, $DE \to B$, because:

$$B = (A \mod 6) \quad C = (A \mod 4) \quad D = (A \mod 3) \quad E = (A \mod 2)$$

no points off for missing $DE \to B$
1 point partial credit for writing just $A \to BCDE$.

(b) (5 points) Using the functional dependencies you have identified at the previous question, decompose the relation in BCNF. You only need to show your final result: the relation names, their attributes, and their keys (underline the key attributes).

Solution: Solution 1:

1. $C^+ = \{C, E\}$. Split $ABCDE$ into CE and $ABCD$
2. $B^+ = \{E, D\}$. Split $ABCD$ into BD and ABC

Final answer: $R1(\underline{A}, B, C)$, $R2(\underline{B}, D)$, $R3(\underline{C}, E)$.
Solution 2:

1. $B^+ = \{B, D, E\}$. Split $ABCDE$ into BDE and ABC

Final answer: $R1(\underline{A}, B, C)$, $R2(\underline{B}, \underline{D, E})$.
$R1(A, B), R2(A, C), R3(A, D), R4(A, E)$ is, strictly speaking, a correct BCNF decomposition, since A is a key and every 2-attribute relation is in BCNF. But it is a poor decomposition because it fails to capture any other functional dependences besides A being a key.

(c) (5 points) The relation $R(A, B, C, D, E)$ satisfies the following functional dependencies:

$$AD \rightarrow E$$
$$CD \rightarrow A$$

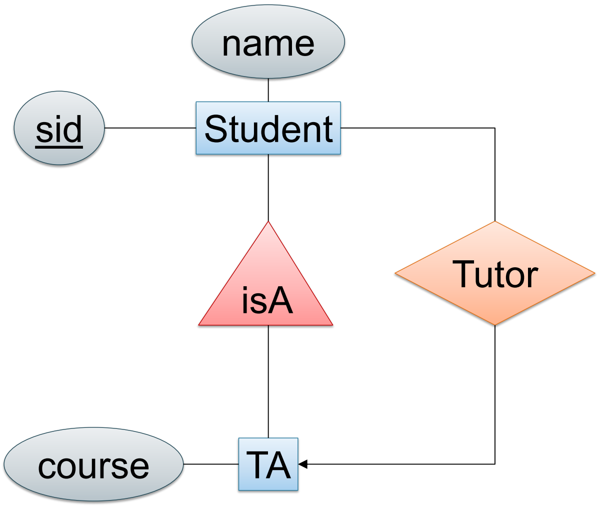
Consider the relation returned by the following query:

```
select distinct R.B, R.D, R.E, 'foo' as F
from R
where R.C = 'bar'
```

Find the key in the resulting table.

Solution: $BD+ = ABCDEF$. Thus, the key is BD. The attributes C and F are determined because they are constants: $C =' bar'$ and $F =' foo'$.

(d) (5 points) Consider the following E/R diagram:



Every student has an ID and a name. Some students are TA's for a course. Every student has at most one Tutor, who is one of the TA's. Write the CREATE TABLE statements to implement the E/R diagram in SQL. Your statements should contain all key and foreign key declarations. Assume that `sid` and `course` are integers, and `name` is a text.

Solution:

```
create table Student(sid int primary key, name text, tutor int references TA);
create table TA(sid int primary key references Student, course int);
```

2 points off for each of: missing key, missing foreign key

1 point off for representing TUTOR in a separate relation

(e) For each statement below indicate whether it is true or false.

i. (3 points) A superkey is any set of attributes X such that $X = X^+$.

i. **False**

True or false?

ii. (3 points) For any set of attributes X, the set X^+ is a superkey.

ii. **False**

True or false?

iii. (3 points) For any set of attributes X, the following identity holds: $(X^+)^+ = X^+$.

iii. **True**

True or false?

iv. (3 points) If $X \cap Y$ is a superkey, then X is also a superkey.

iv. **True**

True or false?

v. (3 points) If $X \cup Y$ is a superkey, then X is also a superkey.

v. **False**

True or false?

7 Transactions

7. (45 points)

(a) A database consists of the following elements:

$$A_1, \dots, A_{1000}$$

The system runs a workload of transactions of the following kind:

```
BEGIN
READ($A_i$)
/* ... compute... compute... compute ... for 0.1 seconds */
WRITE($A_i$)
COMMIT
```

Each transaction reads one element A_i, performs some intensive computation, then writes the same element back. Different transactions may access the same element or different elements.

The system is shared-memory, has 100 CPUs, and uses inter-query parallelism (multiple transactions are run in parallel, but each transaction runs on a single CPU). For serializability, the system uses one lock per element (like SQL Server).[1]

How many transactions per second can the system execute in each case below?

i. (5 points) All transactions access the same element A_1. That is, the transactions update elements in this sequence: $A_1, A_1, A_1, \dots$

Answer Write the number of transactions per second (TPS):

Solution: 10 TPS. The transactions are executed serially.
Almost everyone answered correctly.

[1] While in a real multicore system speedup is affected dramatically by latch contention, our multicore system has magic latches that do not cause any slowdown.

ii. (5 points) The transactions update only elements from the first 50 elements. More precisely, each transaction reads/writes an element A_i chosen uniformly at random from among $A_1, \ldots, A_{50}$. For example, the transactions may update the elements in this sequence: $A_{44}, A_2, A_{13}, A_2, A_{36}, A_{11}, \ldots$
Answer Write the number of transactions per second (TPS):

Solution: 500 TPS. For example, suppose the elements are updated round-robin: $A_1, A_2, \ldots, A_{50}, A_1, A_2, \ldots$ T51 can start only after T1 terminates, which is 0.1s after T1 starts. We run 50 transactions every 0.1s, or 500 TPS. We can sustain this rate because there are 100 CPUs and each can sustain 10 TPS max.
Almost everyone answered correctly.

iii. (5 points) The transactions update all elements. More precisely, each transaction reads/writes an element A_i, choosen uniformly at random from among $A_1, \ldots, A_{1000}$.
Answer Write the number of transactions per second (TPS):

Solution: 1000 TPS, because the system becomes CPU bound. The max rate per CPU is 10 TPS and there are 100 CPUs.
Almost everyone answered correctly.

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

i. (5 points)

$$R_1(A), W_2(B), R_3(C), W_3(A), R_4(D), R_4(B), W_1(D), W_2(C)$$

Solution: Not conflict serializable.
DRAW THE GRAPH

ii. (5 points)

$$R_1(A), W_2(B), R_3(C), W_3(A), R_1(D), R_4(B), W_4(D), W_2(C)$$

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

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

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

i. **False**

True or false?

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

ii. **False**

True or false?

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

iii. **True**

True or false?

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

iv. **False**

True or false?

v. (2 points) When a deadlock occurs, then the database system aborts one or more transactions.

v. **true**

True or false?

(d) A database for course registration maintains a binary element R indicating whether the registration is open or not:

```
R = 1   means registraion is open
R = 0   means registraion is closed
```

For historical reasons, the database has a second element Q representing the same information:

```
Q = 1   means registraion is open
Q = 0   means registraion is closed
```

This is a big problem, because some applications update R, some applications update Q, and some update both. The administrator wrote two transactions T1 and T2 to set them to equal values:

```
set isolation level READ COMMITTED
begin transaction  -- T1
READ(R);
If R=0 Then Q = 0;
       Else Q = 1;
WRITE(Q);
commit
```

```
set isolation level READ COMMITTED
begin transaction  -- T2
READ(Q);
If Q=0 Then R = 0;
       Else R = 1;
WRITE(R);
commit
```

Notice that both transactions run under the READ COMMITTED isolation level. The initial values are $R = 0$ and $Q = 1$. Answer the questions below:

i. (2 points) Is deadlock possible?

i. **Yes**

Yes or no:

ii. (2 points) Is the outcome $R = 0, Q = 0$ possible?

ii. **Yes**

Yes or no:

iii. (2 points) Is the outcome $R = 0, Q = 1$ possible?

iii. **Yes**

Yes or no:

iv. (2 points) Is the outcome $R = 1, Q = 0$ possible?

iv. **No**

Yes or no:

v. (2 points) Is the outcome $R = 1, Q = 1$ possible?

v. **?**

Yes or no: