CSE 344 Final Examination

June 8, 2011, 8:30am - 10:20am

Name: ______________________________

Question	Points	Score
1	20	
2	20	
3	30	
4	25	
5	35	
6	25	
7	20	
8	25	
Total:	200	

- This exam is a closed book exam.
- You have 1h:50 minutes; budget time carefully.
- Please read all questions carefully before answering them.
- Some questions are easier, others harder; if a question sounds hard, skip it and return later.
- Good luck!

1 Basic SQL

1. (20 points)

   Consider a social network database, about people and their relationships. The database has two relations:

   ```
   Person(pid, name)
   Relationship(pid1, rel, pid2)
   ```

   Here `Person.pid` is a key, and `Relationship.pid1` and `Relationship.pid2` are foreign keys; `rel` is a string representing the relation type, and can be `friend` or `enemy`. Note that the relationship is not necessarily symmetric: if `Alice` is friend with `Bob`, this does not imply that `Bob` is friend with `Alice`.

   (a) (10 points) Write the SQL statements that define the relational schema for this database. Assume that `pid`'s are integers, and `name` and `rel` are character strings.

   Answer (write SQL statements):

   Solution:

   ```
   create table Person(
     pid int primary key,
     name varchar(20));
   create table Relationship(
     pid1 int references Person,
     rel varchar(20),
     pid2 int references Person);
   ```

(b) (10 points) Write a **SQL query** that computes, for each person, the total number of their friends. Your query should return results containing the `pid`, the `name`, and the `count`. Note that your query must return exactly one answer for every person in `Person`.

Answer (write a SQL query):

Solution:

```
select x.pid, x.name, count(*)
from Person x left outer join Relationship y
  on x.pid = y.pid1 and y.rel='friend'
group by x.pid, x.name;
```

2 Advanced SQL

2. (20 points)

   Your application needs to enforce the following constraint:

   All my enemies' enemies are my friends

   However, the database administrator has noticed that this constraint does not hold at all ! There are many people whose enemies' enemies are not listed as their friends. Help the database administrator enforce the constraint. You will do this in the following two ways. In each case, write a SQL query that fixes the database according to a certain rule:

   (a) (10 points) The rule is

   My enemies' enemies **shall** *be my friends*

   Answer (write a SQL INSERT query):

   Solution:

```
insert into Relationship
  select x.pid1 as pid1, 'friend' as rel, y.pid2 as pid2
  from Relationship x, Relationship y
  where x.rel = 'enemy' and x.pid2=y.pid1 and y.rel='enemy'
   and not exists (select * from Relationship z
                   where x.pid1=z.pid1 and z.rel='friend'
                     and y.pid2=z.pid2);
```

(b) (10 points) The rule is
I shall make peace with all my enemies whose enemies are not my friends
Answer (write a SQL DELETE query):

Solution:

```
delete from Relationship
where rel='enemy' and
   exists (select *
          from Relationship y
          where pid2=y.pid1 and y.rel='enemy'
           and not exists(select * from Relationship z
                         where pid1=z.pid1 and z.pid2=y.pid2
                           and z.rel = 'friend'));
```

3 Relational algebra, Relational Calculus, Datalog

3. (30 points)

   (a) (15 points) Consider the following relational schema:

   ```
   R(A)
   S(A,B)
   T(B,C)
   U(A,C)
   ```

   The following query is expressed in the relational calculus:

   $$Q(x) = R(x) \wedge \forall y.(S(x,y) \Rightarrow \forall z.(T(y,z) \Rightarrow U(x,z)))$$

   1. Write an equivalent query in non-recursive datalog with negation.

   Solution: We need to be careful and make the query domain independent:

   ```
   K(x,y) :- R(x), T(y,z), not U(x,z)
             /* R(x) added for domain independence */
   L(x)   :- S(x,y), not K(x,y)
   Q(x)   :- R(x), not L(x)
   ```

   2. Write this query in the relational algebra.

   Solution: All joins below are natural joins:

   $$K = \Pi_{AB}((R \bowtie T) - (T \bowtie U))$$
   $$L = \Pi_{A}(S - K)$$
   $$\mathtt{Q} = R - L$$

(b) (15 points) Consider a database consisting of the following two relations:

```
Person(pid, name)
Trusts(pid1, pid2)
```

Answer each question below by writing a query in non-recursive datalog with negation. You answer should return the person id and the name. For example, if the question were *find people who trust everyone except themselves* then your answer would be:

```
S(p)    :- Person(p,n), Person(q,m), not Trusts(p,q), p != q
A(p,n)  :- Person(p,n), not S(p)
```

1. A *loner* is a person who trusts no-one but himself. Return all loners.
   Answer (write a datalog query):

Solution:

```
NA(p)  :- Trusts(p, x), p != x
A(p,n) :- Person(p,n), not NA(p)
```

2. A *loyal* is a person who trusts only those who trust him. Return all loyals.

Solution:

```
NA(p)   :- Trusts(p,x), not Trusts(x,p)
A(p,n)  :- Person(p,n), not NA(p)
```

3. A *ruler* is a person who trusts only those who trust only him. Return all rulers.

Solution:

```
NA(p)  :- Trusts(p,x), Trusts(x,y), p!=y
A(p,n) :- Person(p,n), not NA(p)
```

4 XML/XPath/XQuery

4. (25 points)

   Consider an XML document that contains data about books. Consider the following two XPath queries on this XML data:

```
P1 = /bib/book[@year<2005][editor/last/text()='Samet'][price < 99]/title/text()
P2 = /bib/book[author/last/text()='Hull'][author/last/text()='Vianu']/title/text()
```

   In this problem you are asked to design a relational schema for this data, then translate the two XPath queries into SQL over that schema.

   (a) (10 points) Assume that the XML data conforms to the following DTD:

```
<!DOCTYPE bib [
<!ELEMENT bib  (book* )>
<!ELEMENT book  (title,  (author+ | editor+ ), publisher?, price )>
<!ATTLIST book  year CDATA  #REQUIRED >
<!ELEMENT author  (last, first )>
<!ELEMENT editor  (last, first, affiliation )>
<!ELEMENT title  (#PCDATA )>
<!ELEMENT last  (#PCDATA )>
<!ELEMENT first  (#PCDATA )>
<!ELEMENT affiliation  (#PCDATA )>
<!ELEMENT publisher  (#PCDATA )>
<!ELEMENT price  (#PCDATA )>
]>
```

   1. Design a relational schema for the XML data. Your schema should have relations corresponding to entity sets such as `Book`, `Author`, etc., as well as relationships between these entity sets. Write only the relation names and their columns; for example, `Author(aid, last, first)`; do not write the field types nor the key/foreign key constraints.
      Answer (write a relational schema):

      Solution:

      ```
      Book(bid, title, publisher, price)
      Author(aid, last, first)
      Editor(eid, last, first, affiliation)
      Writes(aid, bid)
      Edits(edi, bid)
      ```

2. Translate the two XPath queries `P1` and `P2` into SQL queries over your relational schema.

```
P1 = /bib/book[@year<2005][editor/last/text()='Samet'][price < 99]/title/text()
P2 = /bib/book[author/last/text()='Hull'][author/last/text()='Vianu']/title/text()
```

Answer (write two SQL queries):

Solution:

```
select Book.title
from book, edits, editor
where book.bid = edits.bid and edits.bid = editor.bid
 and book.year < 2005 and book.price < 99
 and editor.last = 'Samet'

select x.title
from book x, writes y, author z, writes u, author v
where x.bid = y.bid and y.aid = z.aid
  and x.bid = y.bid and y.aid = z.aid
```

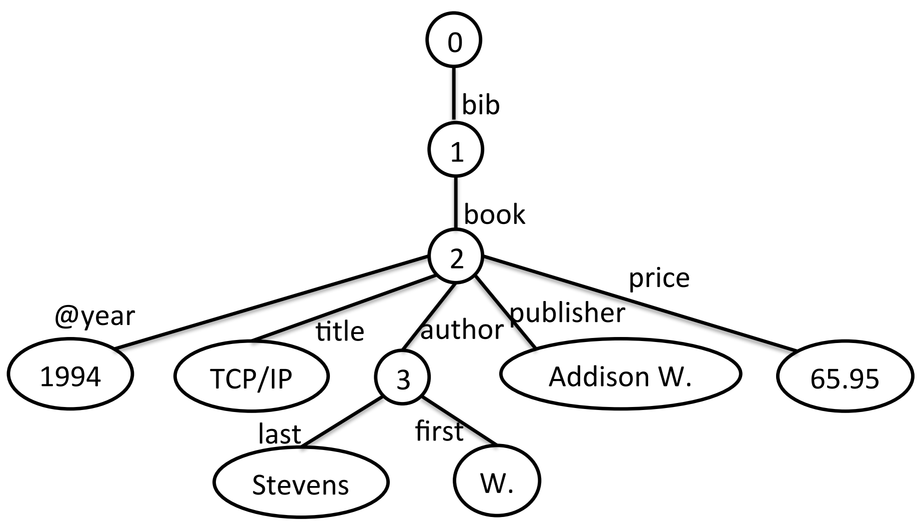
(b) (10 points) Next, assume that no DTD is given, and that we do not know anything about the structure of the XML document. In this case we store it in generic table:

`Element(id, tag, child)`

`id` is a node identifier, `tag` is a tag (or attributes) in that node, `child` is the identifier, or the content of the child. For example, if the document were:

```
<bib> <book year="1994">
        <title>TCP/IP Illustrated</title>
        <author><last>Stevens</last><first>W.</first></author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
      </book>
</bib>
```

Then its tree representation, and tabular representation are:



Element:

id	tag	child
0	bib	1
1	book	2
2	@year	1994
2	title	TCP/IP
2	author	3
2	publisher	Addison Wesley
2	price	65.95
3	last	Stevens
3	first	W.

The root node has always identifier 0, but all others can be arbitrary.

Translate the two XPath queries P1 and P2 into SQL over the table `Element`.

For example, if the XPath query were:

```
P0 = /bib/*/title/text()
```

then your SQL query would be:

```
select z.child
from Element x, Element y, Element z
where x.id = 0 and x.tag = 'bib'
  and x.child = y.id
  and y.child = z.id and z.tag = 'title'
```

Answer write two SQL queries for P1, P2:

```
P1 = /bib/book[@year<2005][editor/last/text()='Samet'][price < 99]/title/text()
P2 = /bib/book[author/last/text()='Hull'][author/last/text()='Vianu']/title/text()
```

Solution: Let E stand for Element

```
select distinct xtitle.child
from E xbib, E xbook, E xeditor, E xlast, E xprice, E xtitle
where xbib.id = 0 and xbib.tag='bib'
  and xbook.id=xbib.child and xbook.tag='book'
  and xeditor.id=xbook.child and xeditor.tag='editor' and xeditor.child='Samet'
  and xprice.id=xbook.child and xprice.tag='price' and xprice.child <'99'
  and xtitle.id=xbook.child and xtitle.tag='title'

select distinct xtitle.child
from E xbib, E xbook, E xa1, E xl1, E xa2, E xl2, E xtitle
where xbib.id = 0 and xbib.tag='bib'
  and xbook.id=xbib.child and xbook.tag='book'
  and xa1.id = xbook.child and xa1.tag = 'author'
  and xl1.id = xa1.child and xl1.tag = 'last' and xl1.child = 'Hull'
  and xa2.id = xbook.child and xa2.tag = 'author'
  and xl2.id = xa2.child and xl2.tag = 'last' and xl2.child = 'Vianu'
  and xtitle.id = xbook.id and xtitle.tag = 'title'
```

(c) (5 points) Consider the advantages and disadvantages of using the two relational representations of the XML data. For each of the criteria below, write a + if one representation is better, write a − if it is worse, and write an = is there is no clear advantage. For illustration, the first two answers are already filled out.

Criteria	Representation	
	DTD-based	DTD-free
Uses fewer tables	−	+
Uses less space	=	=
Easy to make changes to the structure (DTD)		
The XPath query `//*` translates to a simple SQL query		
Typical XPath queries translate to simple SQL queries		
Easy to check that the data is a tree		

5 Conceptual Design, Constraints, Views

5. (35 points)

(a) (10 points) Consider a relation $R(A, B, C, D, E)$ that satisfies the following functional dependencies:

$$ABC \to D$$
$$E \to B$$
$$AD \to C$$

Decompose the schema in BCNF. Show all your steps.

Answer (Show the steps leading to the BCNF decomposition):

Solution: There are two solutions:

Solution 1:

Table	$X^+ = ?$	New table 1	New table 2
$R(A, B, C, D, E)$	$ABC+ = ABCD$	$R_1(A, B, C, D)$	$R_2(A, B, C, E)$
$R_1(A, B, C, D)$	$AD+ = ACD$	$R_3(A, C, D)$	$R_4(A, B, D)$
$R_2(A, B, C, E)$	$E+ = BE$	$R_5(B, E)$	$R_6(A, C, E)$

Answer: $R_3(A, C, D), R_4(A, B, D), R_5(B, E), R_6(A, C, E)$.

Solution 2:

Table	$X^+ = ?$	New table 1	New table 2
$R(A, B, C, D, E)$	$E+ = BE$	$R_1(B, E)$	$R_2(A, C, D, E)$
$R_2(A, C, D, E)$	$AD+ = ACD$	$R_3(A, C, D)$	$R_4(A, D, E)$

Answer: $R_1(B, E), R_3(A, C, D), R_4(A, D, E)$.

(b) (5 points) For each statement below, indicate whether it is true or false. You do not need to justify your answer.

1. A materialized view may contain data that is not up to date.

   (b) **true**

   True or false ?.

2. A query that uses a virtual view always runs much faster than the same query using a materialized view.

   (b) **false**

   True or false ?.

3. An index is special case of a virtual view.

   (b) **false**

   True or false ?.

4. An index is a special case of a materialized view.

   (b) **true**

   True or false ?.

5. It takes much longer to create a virtual view than a materialized view.

   (b) **false**

   True or false ?.

(c) (10 points) Consider the table below:

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_3	c_1
a_2	b_3	c_2

For each of the functional dependencies listed below, indicate whether it holds or not. If it holds, write OK. If it does not hold, indicate two tuples in the table above that violate the functional dependency. Refer to the tuples as 1,2,3,4; for example, you may say that $A \to C$ fails because of the tuples 3,4.

FD	Holds ?
$B \to A$	
$C \to A$	
$A \to B$	
$C \to B$	
$A \to C$	
$B \to C$	
$BC \to A$	
$AC \to B$	
$AB \to C$	

Solution:

FD	Holds ?
$B \to A$	holds
$C \to A$	fails: tuples 1,3
$A \to B$	fails: tuples 1,2
$C \to B$	fails: tuples 1,3
$A \to C$	fails: tuples 1,2
$B \to C$	fails: tuples 3,4
$BC \to A$	holds
$AC \to B$	holds
$AB \to C$	fails: tuples 3,4

(d) (10 points) Consider two relations `R(A,B)`, `S(C,D,E)`, with the following functional dependencies. In R: $A \to B$. In S: $C \to D$. Consider the following view definition:

```
create view V as
  select distinct R.A, R.B, S.D, S.E
  from R, S
  where R.B=S.C and S.E=55
```

For each of the following functional dependencies below, indicate whether they hold in the view $V(A, B, D, E)$:

1. $A \to D$.

   (d) **true**

   True or false ?

2. $E \to B$.

   (d) **false**

   True or false ?

3. $A \to E$.

   (d) **true**

   True or false ?

4. $D \to A$.

   (d) **false**

   True or false ?

6 Transactions

6. (25 points)

Consider a database consisting of a single relation R:

R:

A	B
1	10
2	20

Two transactions run concurrently on this database, resulting in the following schedule:

Line	T1	T2
1	begin;	
2		begin;
3	update R set B = (select sum(B) from R) where A=1;	
4		update R set B = (select sum(B) from R) where A=2;
5	select * from R;	
6		select * from R;
7	insert into r values (3,300);	
8		insert into r values (4,400);
9	select * from R;	
10		select * from R;
11	update R set B = (select sum(B) from R) where A=1;	
12		update R set B = (select sum(B) from R) where A=2;
13	select * from R;	
14		select * from R;
15	commit;	
16		commit;

(a) (5 points) Is this schedule possible in SQL Lite ? If not, then indicate the first line where SQL Lite will change the schedule.

(a) **No: line 4**

Yes ? Or No (and indicate line number) ?

(b) (5 points) Is this schedule possible in Postgres ? If not, then indicate the first line where Postgres will change the schedule.

(b) **Yes**

Yes ? Or No (and indicate line number) ?

(c) (10 points) Consider running these two transactions in Postgres, using isolation level SERIALIZABLE. Assuming postgres executes exactly the schedule above, indicate the result of each of the six `select *` statements, as well as the content of the table after both transactions commit.

Line Number	Result of `select * from r;`
5	
6	
9	
10	
13	
14	
after both commit	

Solution:

Line Number	Result of `select * from r;`
5	(1,30), (2,20)
6	(1,10), (2,30),
9	(1,30), (2,20), (3,300)
10	(1,30), (2,20), (4,400)
13	(1,350), (2,20), (3,300)
14	(1,10), (2,440), (4,400)
after both commits	(1,350), (2,440), (3,300), (4,400)

(d) (5 points) Is the schedule above serializable ?

(d) **NO**

Yes or No ?

7 Parallel Data Processing

7. (20 points)

(a) (5 points) You work for an Information company, and your job is to manage a 1 TB dataset. You have a large database with 100 servers, and there is a job that runs every day and takes 10 hours to complete. You try to explain to your boss the concepts of speedup and scaleup. Help your boss understand these concepts by illustrating on your databases how and ideal speedup, and an ideal scaleup would work:

	Number of Processors	Database Size	Running time
Current	100	1 TB	10 hours
Ideal speedup:			
Ideal scaleup:			

(b) (5 points) You are the salesperson of a database server company, and you are offering three kinds of parallel databases: (A) shared memory, (B) shared disk, and (C) shared nothing. Your customer is interested in the following criteria: speedup, scaleup, cost per processing unit, and ease of programming. Help your customer understand the pros and cons of these architecture by filling out the table below, with signs $-$ (poor), $=$ (neutral), or $+$ (good).

	(A) Shared memory	(B) Shared disk	(C) Shared nothing
Speedup			
Scaleup			
Cost / processing unit			
Ease of programming			

(c) (10 points) We have a large table $R(A, B)$ where A is an integer attribute. The table has 1 Billion ($= 10^9$) records, and the values of A are exactly the first 1 Billion perfect squares. That is, $R.A$ is $1, 4, 9, 16, \ldots, 10^{18}$. We have a parallel database with only two servers, $P = 2$, and wish to partition the table R among these servers. We consider two partition strategies: hash based partition, where the hash function is $h(A) = 1 + (A \texttt{ mod } 2)$, and range partition, where the two ranges are $[0, 0.5 \cdot 10^{18}]$ and $(0.5 \cdot 10^{18}, 10^{18}]$. Compute in each case the number of tuples stored on each server:

	# of tuples on Server 1	# of tuples on Server 2
Hash-partition		
Range partition		

8 Finding Similar Items

8. (25 points)

   (a) (5 points) Consider the following two words:

   `s1 = tamper`
   `s2 = teleporter`

   1. Compute the edit distance between the two strings. Assume that we allow three edit operations: insert, replace, delete, and each has cost 1. You may use the table below to compute the answer, but you do not need to fill in the entire table: you only need to write the final answer.

		t	a	m	p	e	r
t							
e							
l							
e							
p							
o							
r							
t							
e							
r							

(a) ______________

The edit distance is:

   2. Represent both s1 and s2 as sets of 2-grams (`tamper` has 5 2-grams,a nd `teleporter` has 9 2-grams). Compute their Jaccard similarity.

(a) ______________

When $q = 2$, the Jaccard similarity is:

   3. Represent both s1 and s2 as sets of 3-grams. Compute their Jaccard similarity.

(a) ______________

When $q = 3$, the Jaccard similarity is:

(b) (20 points) You have a private collection of 1M documents $d_1, d_2, \dots, d_n$, $n = 10^6$. You also crawled the Web and collected 100B documents $w_1, w_2, \dots, w_N$, $N = 10^{11}$. You would like to find potential copyright violations (suspected copies of your private documents). Concretely, you want to find all pairs of documents (d_i, w_j) that have a Jaccard similarity ≥ 0.8:

$$J(d_i, w_j) \geq 0.8$$

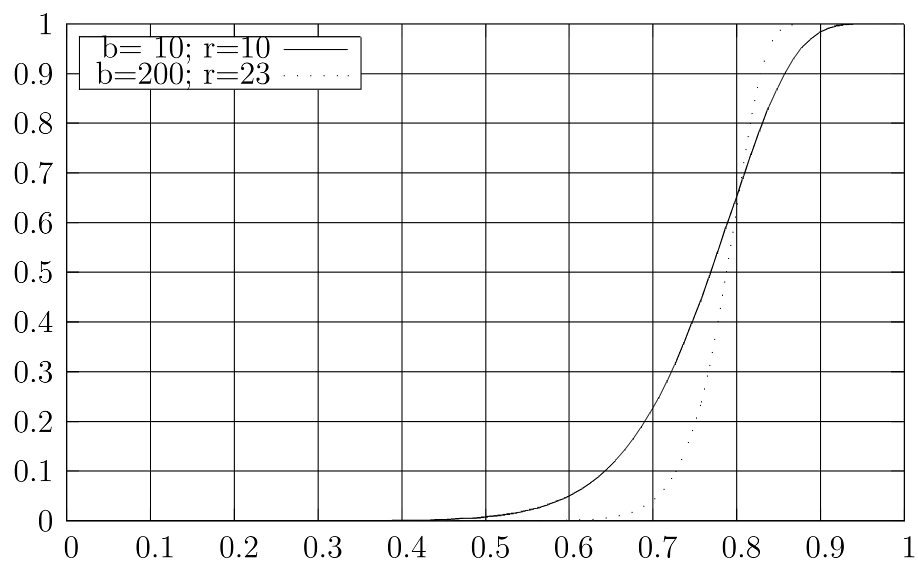
Brute-force would require that you compare $n \cdot N = 10^{17}$ pairs of documents. Instead of brute-force, you are using minhashes and LSH. The hash function that you use for LSH returns 10 bytes.

You consider two options:

Option 1 Use $m = 100$ minhashes. Split them into $b = 10$ bands of $r = 10$ minhashes each.

Option 2 Use $m = 4300$ minhashes. Split them into $b = 200$ bands of $r = 23$ minhashes each.

Recall that each band results in one hash value, which has 10 bytes in our case. For both options the inflexion point is at $\left(\frac{1}{b}\right)^{\frac{1}{r}} \approx 0.8$, see the graph below:



As usual, your algorithm based on LSH has four steps:

Step 1 Compute and store the LSH of all $n = 10^6$ private documents. (Recall that you need to store b hash values of 10 bytes each, per document.)

Step 2 Compute and store the LSH for all $N = 10^{11}$ Web documents.

Step 3 Find all pairs of documents (d_i, w_j) that have a common hash value.

Step 4 For each pair (d_i, w_j) of documents returned during step 3, compute the Jaccard similarity; if it is ≥ 0.8, then return the pair (d_i, w_j).

You expect the following:

- Approximatively 10,000 pairs (d_i, w_j) have $J(d_i, w_j) = 0.85$. (If Step 3 fails to return such a pair, then it is called a *false negative*.)
- Approximatively 10^{10} pairs (d_i, w_j) have $J(d_i, w_j) = 0.6$. (If Step 3 returns such a pair, then it is called a *false positive*.)
- The rest of the 10^{17} pairs (d_i, w_j) have a very small Jaccard similarity; we will assume that $J(d_i, w_j) = 0$.

Compute the cost of each option below. Use the graph to estimate the cost; you only need to give approximate answers. (If the graph gives a number that is to small to read, write *negligible*.)

Cost	Option 1	Option 2
Storage used by Step 1		
Storage used by Step 2		
# Total pairs returned by Step 3		
# False negatives		

Solution:

Cost	Option 1	Option 2
Storage used by Step 1	$100 \cdot 10^6 = 10^8$	$2000 \cdot 10^6 = 2 \cdot 10^9$
Storage used by Step 2	$100 \cdot 10^{11} = 10^{13}$	$2000 \cdot 10^{11} = 2 \cdot 10^{14}$
# Total pairs returned by Step 3	$5 \cdot 10^8$	negligible
# False negatives	1000	100 (or negligible)