

CSE 344 Midterm

Friday, February 8, 2013, 9:30-10:20

Name: _____

Question	Points	Score
1	60	
2	30	
3	10	
Total:	100	

- This exam is open book and open notes but NO laptops or other portable devices.
- You have 50 minutes; budget time carefully.
- Please read all questions carefully before answering them.
- Some questions are easier, others harder. Plan to answer all questions, do not get stuck on one question. If you have no idea how to answer a question, write your thoughts about the question for partial credit.
- Good luck!

1 SQL

1. (60 points)

A search engine company maintains a database with the following schema:

QueryLog(uid,qid,word)

User(uid, name, language)

Dictionary(word, language)

QueryLog is the archive of all keyword queries issued by the users; User is the set of registered users; Dictionary stores some common words in several languages. Note that one word may occur in more than one language.

To illustrate, we consider the following search queries:

```

Joe:      valentine day
Jaque:    day trip
Joe:      sidereal day
Jim:      day
  
```

and the following small database:

QueryLog:

uid	qid	word
u1	q1	valentine
u1	q1	day
u2	q2	day
u2	q2	trip
u1	q3	sidereal
u1	q3	day
u3	q4	day

User:

uid	name	language
u1	Joe	English
u2	Jaque	French
u3	Jim	English

Dictionary:

word	language
valentine	Latin
valentine	English
day	English
trip	English
sidereal	Latin
cosmos	Greek

```
QueryLog(uid,qid,word)
User(uid,name, language)
Dictionary(word, language)
```

- (a) (15 points) Write a SQL query that returns the top 10 most common queried words, in decreasing order of their frequency. For our small example, the query would return:

day	4
sidereal	1
trip	1
valentine	1

Recall that you can limit the number of tuples returned using `limit`, for example:

```
select *
from T
limit 20
```

returns only 20 answers.

```
QueryLog(uid,qid,word)
User(uid,name, language)
Dictionary(word, language)
```

- (b) (15 points) Write a SQL query that returns all languages that were never used in a query. In our small example, your query would return **Greek**, because no user asked for any word in **Greek**.

```
QueryLog(uid,qid,word)
User(uid,name, language)
Dictionary(word, language)
```

- (c) (15 points) You want to identify users who speak a foreign language. We assume that a user speaks a language L if she issues at least one query Q where all words are in L ; if L is different from the user's native language (stored in `Users`) then we assume that she speaks the foreign language L . Write a query that returns all users that speak a foreign language. Your query should return the `uid`, the user name, and the foreign language.

For example, in our toy database your query should return `Jaque`, because he issued the query `day trip` where all words are in `English`; you should not return `Joe`: for example in his query `sidereal day`, while the first word is in Latin, the second is only in his native `English`.

- (d) (10 points) The company monitors some users, and some keywords on a regular basis. That is, the company frequently runs queries of the following kind (where the values “Joe” and “valentine” are replaced with different constants)

QueryType1:

```
select y.qid, y.word
from User x, QueryLog y
where x.uid = y.uid and x.name = 'Joe'
order by y.qid, y.word;
```

QueryType2:

```
select x.uid, x.name
from User x, QueryLog y
where x.uid = y.uid and y.word = 'valentine';
```

To speedup queries like these (where Joe and valentine are replaced with various constants), the database administrator considers creating the following indexes:

```
create index QL1 on QueryLog(uid)
create index QL2 on QueryLog(word)
create index QL3 on QueryLog(uid,word)
```

```
create index U1 on User(uid)
create index U2 on User(name)
create index U3 on User(uid,name)
```

Help the database administrator choose which indexes to create, by filling in the table below. For each entry, indicate one of the following tree options:

OK: this index may speed up queries of this type.

Dominated by X: the index may speed up queries of this type, but the index X would be at least as good, so the DBA should always prefer X.

Bad: the index is never useful for queries of this type, the DBA should not even consider it.

Fill in with one of OK, Dominated by X, Bad:

	QueryType1 <pre>select y.qid, y.word from User x, QueryLog y where x.uid = y.uid and x.name = 'Joe' order by y.qid, y.word;</pre>	QueryType2 <pre>select x.uid, x.name from User x, QueryLog y where x.uid = y.uid and y.word = 'valentine';</pre>
QL1 QueryLog(uid)		
QL2 QueryLog(word)		
QL3 QueryLog(uid, word)		
U1 User(uid)		
U2 User(name)		
U3 User(uid, name)		

(e) Answer the questions below:

i. (1 point) Which of the following is *not* a database management system:

- (a) Oracle
- (b) SQL Server
- (c) Excel
- (d) SQLite
- (e) Postgres

i. _____

Answer (a), (b), (c), (d) or (e).

ii. (4 points) Joe Pythonson never took a course in databases, but he is a proficient python programmer. For his job, he needs to develop an address book application, which stores names, phone numbers, emails, and addresses. Joe plans to develop it entirely in python, without using any database management system. You think that's a terrible idea, and try to convince Joe to use a database system like SQL Server or postgres. How would you answer to Joe's arguments?

(a) Joe: "I can implement the access procedure to the phone book very efficiently! I know splay-trees very well, they are efficient and I can implement them quickly". Your answer:

1. "Splay-trees do not support joins".
2. "You need to make splay trees persistent, and extend them with concurrent access: to implement them you will need much more time than you think".
3. "Splay trees are copyrighted, and you cannot use them in your commercial application"

ii. _____

1, 2, or 3?

- (b) Joe: “If I develop everything in python then the entire address book is in a single process. Thus, my application will run faster than, say, using SQL Server, which is a client server configuration and requires a network connection for every single user request”. Your answer:
1. “True, but: the client-server configuration has the advantage of supporting simultaneous access by multiple users. If you program in python, then you have to implement multi users access yourself, and this requires a lot of work”.
 2. “False: the client-server configuration will run queries much more efficiently than the single process implementation because the database system optimizes the queries before running them.”
 3. “True, but: the client-server configuration can scale to much larger datasets, because the data is in a different process.”

ii. _____

1, 2, or 3?

- (c) Joe: “I have just invented a new and amazing compression method for email addresses (for which I expect to receive the Turing Award). Using this specialized email compression technique I can make the email lookup function run 10^6 times faster than any database system, crushing my competitors”. Your answer:
1. “You don’t need to give up on a database management system: get the source code of an open source database management system, like postgres, implement your amazing email compression technique, recompile it, and still use your specialized method to crush the competition”.
 2. “Users will be turned away from your address book application when they learn that it is not powered by a database management system, even with your ridiculously fast lookup”.
 3. “This only speeds up access to email addresses, while all other accesses will be have the same speed; users won’t notice much difference, while you will put a lot of effort reimplementing much of the functionality offered by a database management system.”

ii. _____

1, 2, or 3?

2 Relational Algebra, Datalog, and Relational Calculus

2. (30 points)

Consider the following database schema:

```
Person(name)
Friend(name1, name2)
```

Person contains all persons in the database, while **Friend** contains all friendship relationships.

(a) (10 points) The SQL query below counts how many persons have no friends in common with Alice:

```
select count(*)
from Person x
where not exists
    (select *
     from Friend y, Friend z
     where x.name =y.name1
          and z.name1='Alice'
          and z.name2=y.name2);
```

Write a relational algebra expression that is equivalent to this SQL query. Write your answer as query plan (i.e. a tree).

Person(name)
Friend(name1, name2)

(b) (10 points) Write a query in the relational calculus that finds all persons who are friends with all of Alice's friends.

(c) (10 points) Write a query in datalog with negation that finds all persons who are friends with all of Alice's friends.

3 XML and XPath

3. (10 points)

Consider the following XML document:

```
<persons>
  <person>
    <name> Alice </name>
    <friend>Bob</friend>
    <friend>Carol</friend>
  </person>
  <person>
    <name> Bob </name>
    <friend>Carol</friend>
  </person>
  <person>
    <name>Carol</name>
    <enemy>Alice</enemy>
  </person>
</persons>
```

(a) (5 points) Draw the tree representation of the XML document.

(b) (5 points) Represent the same data in relational format. Show the tables, including the table names, their attributes, and their values (tuples).