CSE 414 Midterm

Friday, April 29, 2016, 1:30-2:20

Name: ______________________________

Question	Points	Score
1	50	
2	20	
3	30	
Total:	100	

- This exam is CLOSED book and CLOSED devices.
- You are allowed ONE letter-size page with notes (both sides).
- 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 and Indexing

1. (50 points)

   Consider the following database an online video rental service:

   ```
   Customer(cid, name, country)
   Movie(mid, title, year)
   Rent(cid, mid, date)
   Rate(cid, mid, score)
   ```

   The `Customer` relation store all customers of the rental service, their names and the countries where they are located.

   `Movie` stores all movies available for rent, their titles and the year of production.

   When a customer rents a movie, a record is inserted in the relation `Rent`; a customer is allowed to rent a movie only once.

   After watching the movie, a customer is allowed to rate the movie, with a score from 1 (worst) to 5 (best): this information is stored in the `Rate` table. Notice that a customer may rate a movie only if she/he has also rented the movie.

   All primary keys are underlined. The attributes' types are as follows:

   - `cid, mid, year, score` are integers.
   - `name, country, title, date` are text.

CONTINUED ON NEXT PAGE

```
Customer(cid, name, country)
Movie(mid, title, year)
Rent(cid, mid, date)
Rate(cid, mid, score)
```

(a) (10 points) Write SQL statements to create the tables for the rental service application. Make sure you choose the right types for the attributes, and define all key and foreign key constraints.

Solution:

```
drop table if exists Rent;
drop table if exists Rate;
drop table if exists Movie;
drop table if exists Customer;

create table Customer(cid int primary key, name text, country text);
create table Movie(mid int primary key, title text, year int);
create table Rent
  (cid int references Customer,
   mid int references Movie,
   date text,
   primary key (cid, mid));
create table Rate
  (cid int, mid int, score int,
   primary key (cid, mid),
   foreign key (cid, mid) references Rent);
```

```
Customer(cid, name, country)
Movie(mid, title, year)
Rent(cid, mid, date)
Rate(cid, mid, score)
```

(b) (10 points) The *Canadian rating* of a movie is the average score received by that movie from customers in Canada. Write a SQL query that computes the average Canadian rating of each movie; order your answers by the Canadian rating, starting from the highest rating.

Solution:

```
select x.mid, x.title, avg(y.score) as s
from Movie x, Rate y, Customer z
where x.mid = y.mid and y.cid = z.cid
  and z.country = 'Canada'
group by x.mid, x.title
order by s desc;
```

1 point off for missing/wrong group by
1 point off for missing/wrong order by
1 point off for missing/wrong avg
2-3 points off for nested queries
2 points off for missing joins (e.g. no Rate)
1 point off for missing/wrong attributes in SELECT
2 points off for missing test for Canada (usually because of missing Customer table)

No points were taken off for redundant join with Rent
No pionts off for missing Movie table (i.e. OK to return just y.mid)

```
Customer(cid, name, country)
Movie(mid, title, year)
Rent(cid, mid, date)
Rate(cid, mid, score)
```

(c) (10 points) We want to return the `name` of all customers who were never rented a movie made before 2010; in other words, we want to find customers who rented only movies made in 2010 or later. Indicate which of the query or queries below answers this question correctly:

```
Q1 = select distinct x.name
     from Customer x, Rent y
     where x.cid = y.cid
       and not exists (select *
                       from Movie z
                       where y.mid = z.mid
                         and z.year < 2010);

Q2 = select distinct x.name
     from Customer x
     where not exists (select *
                       from Rent y, Movie z
                       where x.cid = y.cid
                         and y.mid = z.mid
                         and z.year < 2010);

Q3 = select distinct x.name
     from Customer x
     where not exists (select *
                       from Rent y
                       where x.cid = y.cid
                        and not exists
                            (select *
                             from Movie z
                             where y.mid = z.mid
                               and z.year >= 2010));

Q4 = select distinct x.name
     from Customer x left outer join Rent y on x.cid = y.cid
       left outer join Movie z on y.mid = z.mid and z.year < 2010
     group by x.name
     having count(z.mid) = 0;
```

(c) **Q2, Q3, Q4**

Answer with any subset of Q1, Q2, Q3, Q4:

```
Customer(cid, name, country)
Movie(mid, title, year)
Rent(cid, mid, date)
Rate(cid, mid, score)
```

(d) (10 points) Write a SQL query that returns the cid's and names of all customers who rented all the movies where "Alice" gave a score of 5. For example, if Alice gave a score of 5 to "Mad Max" and "Hunger Games" and gave no other score of 5, then your query should return the customers who rented both "Mad Max" and "Hunger Games".

Solution:

```
select *
from Customer x
where not exists
    (select *
     from Customer a, Rate b, Movie z
     where a.cid = b.cid and b.mid = z.mid
       and a.name = 'Alice' and b.score = 5
       and not exists (select * from Rent y
                       where x.cid = y.cid and y.mid = z.mid));
```

4 points off for one missing negation
8 points off for both missing negations
3 points off for wrong order of negated subqueries

```
Customer(cid, name, country)
Movie(mid, title, year)
Rent(cid, mid, date)
Rate(cid, mid, score)
```

(e) Consider the following indexes:

```
create clustered index C_cid on Customer(cid);
create index C_name on Customer(name);
create index C_country on Customer(country);

create clustered index M_mid on Movie(mid);
create index M_year on Movie(year);

create index R_cid on Rate(cid);
create index R_mid on Rate(mid);
create index R_score on Rate(score);
```

For each query below, indicate which indexes might be used by the query optimizer in order to improve query performance. You may answer with more than one; if no index may be used, then write NONE. As usual, the optimizer makes uniformity assumptions when estimating if an index is useful.

i. (1 point) Which indexes might be used by this query?

```
select distinct x.country
from Customer x
where x.name = 'Alice';
```

Solution: `C_name`

ii. (3 points) Which indexes might be used by this query?

```
select distinct z.title
from Customer x, Rate y, Movie z
where x.cid = y.cid and y.mid = z.mid
  and x.name = 'Alice' and y.score = 5;
```

Solution: `C_name,R_cid, M_mid`

iii. (3 points) Which indexes might be used by this query?

```
select distinct x.name
from Customer x, Rate y, Movie z
where x.cid = y.cid and y.mid = z.mid
  and y.score = 5 and z.year = '1910';
```

Solution: `M_year, R_mid, C_cid`

iv. (3 points) Which indexes might be used by this query?

```
select distinct x.name
from Customer x, Rate y, Movie z
where x.cid = y.cid and y.mid = z.mid
  and x.country = z.title and y.score = 5;
```

Solution: NONE

2 Relational Algebra and Datalog

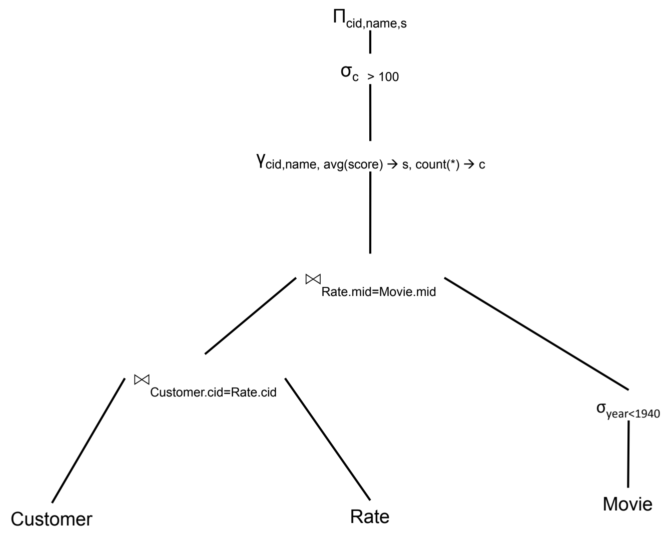
2. (20 points)

   Consider the same relational schema as before:

   ```
   Customer(cid, name, country)
   Movie(mid, title, year)
   Rent(cid, mid, date)
   Rate(cid, mid, score)
   ```

   (a) (5 points) Write a Relational Algebra expression in the form of a logical query plan (i.e., draw a tree) that is equivalent to the SQL query below. Your query plan does not have to be necessarily "optimal": however, points will be taken off for overly complex solutions.

   ```
   select x.cid, x.name, avg(y.score) as s
   from Customer x, Rate y, Movie z
   where x.cid = y.cid and y.mid = z.mid and z.year < 1940
   group by x.cid, x.name
   having count(*) > 100;
   ```


$\Pi_{cid,name,s}$
$\sigma_{c > 100}$
$\gamma_{cid,name, avg(score) \rightarrow s, count(*) \rightarrow c}$
$\bowtie_{Rate.mid=Movie.mid}$
$\bowtie_{Customer.cid=Rate.cid}$
$\sigma_{year<1940}$
Customer
Rate
Movie

Solution:

```
Customer(cid, name, country)
Movie(mid, title, year)
Rent(cid, mid, date)
Rate(cid, mid, score)
```

(b) (10 points) Write a query in datalog that returns all customers who rated the movie "Hunger Games" with a score of 3; return their cid's and their names.

Solution:

```
answer(cid, name) :- Customer(cid, name, country),
                     Rate(cid, mid, 3),
                     Movie(mid, 'Hunger Games',year)
```

```
Customer(cid, name, country)
Movie(mid, title, year)
Rent(cid, mid, date)
Rate(cid, mid, score)
```

(c) (5 points) Write a query in datalog with negation that returns all movies who were never rented by any customer living in Canada; return their mid's and their titles.

Solution:

```
NonAnswer(mid) :- Customer(cid, name, 'Canada'),
                  Rent(cid, mid, date),
                  Movie(mid, title, year)
Answer(mid, title) :- Movie(mid, title, year), not nonAnswer(mid)
```

3 Relational Data Model and Query Evaluation

3. (30 points)

   Answer each question below.

   (a) (2 points) True or false? If an attribute of a relation is a foreign key, then all records in the relation must have distinct values for that attribute.

   (a) **False**

   True or false?

   (b) (2 points) In SQL, a relation can have at most one key.

   (b) **True**

   True or false?

   (c) (2 points) In SQL, a relation can have at most one foreign key.

   (c) **False**

   True or false?

   (d) (2 points) The goal of a foreign key is to help the query optimizer evaluate queries more efficiently.

   (d) **False**

   True or false?

(e) (2 points) If the relation $R(A, B)$ has m tuples, and the relation $S(B, C)$ has n tuples, what is the largest possible size of the output of the natural join $R \bowtie S$? Choose one of the following:

(a) m
(b) n
(c) mn
(d) $m + n$
(e) $\max(m, n)$
(f) $(m + n)/2$
(g) None of the above.

(e) **c**

Answer a-g:

(f) Consider two relations $R(A, B), S(C, D)$.

i. (1 point) Is the size of $R \bowtie_{A=C} S$ always larger than or equal to the size of $R \bowtie_{A=C \text{ and } B=D} S$?

i. **Yes**

Yes or no?

ii. (1 point) Is the size of $R \bowtie_{A=C \text{ and } B \neq D} S$ always larger than or equal to the size of $R \bowtie_{A=C \text{ and } B=D} S$?

ii. **No**

Yes or no?

iii. (1 point) Are these two expressions equivalent $\sigma_{A=5}(R \bowtie_{B=C} S) = (\sigma_{A=5}(R)) \bowtie_{B=C} S$?

iii. **Yes**

Yes or no?

iv. (1 point) Are these two expressions equivalent $(R \times S) - (R \bowtie_{B=C} S) = R \bowtie_{B \neq C} S$

iv. **Yes**

Yes or no?

(g) (2 points) True or false? A "disk block" is a mechanical device that moves over the disk platter in order to read from or write to disk.

(g) **False**

True or false?

(h) Assuming $B(R) = 1000$, $T(R) = 200,000$ and $V(R, A) = 500$, estimate the number of disk I/O's for an index-based selection for $\sigma_{A=55}(R)$ in each of the following cases:

i. (2 points) The system uses an unclustered index on $R.A$.

i. **400**

Number of I/O's:

ii. (2 points) The system uses a clustered index on $R.A$.

ii. **2**

Number of I/O's:

(i) (2 points) The main advantage of a clustered index over an unclustered index is that the clustered index uses less space.

(i) **False**

True or false?

(j) (2 points) The main reason why we cannot create too many indexes is because they will slow down updates to the database.

(j) **True**

True or false?

(k) (2 points) There is an important distinction between logical operators and physical operators. Indicate which of the operators below are physical operators:

(a) Theta join
(b) Hash join
(c) Natural join
(d) Eq join
(e) Index-based join
(f) Merge join

(k) **b,e,f**

Answer a-f (choose all that apply):

(l) (2 points) Did the speed of sequential reads from disk increase significantly over the years?

(l) **Yes**

Yes or no?

(m) (2 points) Did the speed of random reads from disk increase significantly over the years?

(m) **No**

Yes or no?