CSE 344 Midterm

Monday, November 9th, 2015, 9:30-10:20

Name: ____________________________

Question	Points	Score
1	30	
2	30	
3	10	
Total:	70	

- 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. (30 points)

   Consider the following database for a summer camp:

   ```
   Camper(cid,name)
   Team(gid,age)
   Assignment(cid,gid)
   ```

   Relation `Camper` contains a tuple for each camper with name `name` and unique id `cid`. Each tuple in relation `Team` holds the unique id of a team and the age of the campers assigned to that team. Finally, table `Assignment` captures the assignments of campers to teams. Primary keys are underlined.

   (a) (10 points) For each query below, indicate if the query correctly returns the `name` of all campers who have not yet been assigned to a team. Anywhere between zero and all queries compute the correct answer. Each query is syntactically correct.

   1.
   ```
   SELECT C.name
   FROM Camper C LEFT OUTER JOIN Assignment A
   ON C.cid = A.cid
   WHERE A.gid IS NULL
   ```

   2.
   ```
   SELECT C.name
   FROM Camper C
   WHERE C.cid NOT IN (SELECT A.cid FROM Assignment A)
   ```

   3.
   ```
   SELECT C.name
   FROM Camper C INNER JOIN Assignment A
   ON C.cid = A.cid
   GROUP BY C.cid, C.name
   HAVING count(*) = 0
   ```

   Solution: (1) and (2)

```
Camper(cid,name)
Team(gid,age)
Assignment(cid,gid)
```

(b) (10 points) Write a SQL query that computes the number of campers in each age group but only for age groups below 9 years old. Note that multiple teams can have campers with the same age.

Solution: The following are two possible solutions:

```
SELECT T.age, COUNT(A.cid)
FROM Team T LEFT OUTER JOIN Assignment A
ON A.gid = T.gid
WHERE T.age < 9
GROUP BY T.age
```

```
SELECT T.age, COUNT(A.cid)
FROM Team T LEFT OUTER JOIN Assignment A
ON A.gid = T.gid
GROUP BY T.age
HAVING T.age < 9
```

```
Camper(cid,name)
Team(gid,age)
Assignment(cid,gid)
```

(c) (10 points) Consider the following query. For each index below, indicate if it can potentially speed up the query if it is the only index that exists.

```
SELECT C.name
FROM Camper C, Team T, Assignment A
WHERE C.cid = A.cid
AND T.gid = A.gid
AND T.age > 6
```

1. Index on `Camper(cid)`
2. Index on `Camper(name)`
3. Index on `Camper(cid,name)`
4. Index on `Camper(name,cid)`
5. Index on `Team(gid)`
6. Index on `Team(age)`
7. Index on `Team(gid,age)`
8. Index on `Team(age,gid)`

Solution:

1. Yes
2. No
3. Yes
4. No
5. Yes
6. Yes
7. Yes
8. Yes

2 Relational Algebra, Datalog, and Relational Calculus

2. (30 points)

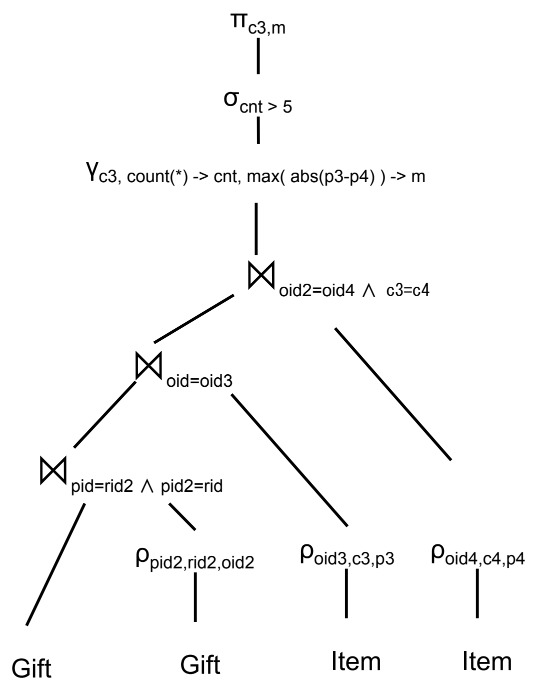
Consider the following database schema. Relation `Item` lists objects with their unique id (`oid`), category, and price. Relation `Gift` has one tuple for every person `pid` that offered a gift to a recipient `rid`. `Gift.oid` references `Item.oid`.

```
Item(oid,category,price)
Gift(pid, rid, oid)
```

(a) (10 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.

```
SELECT O1.category, MAX(ABS(O1.price - O2.price))
FROM Gift G1, Gift G2, Item O1, Item O2
WHERE G1.pid = G2.rid
AND G2.pid = G1.rid
AND O1.oid = G1.oid
AND O2.oid = G2.oid
AND O1.category = O2.category
GROUP BY O1.category
HAVING count(*) > 5
```

Solution:


$\pi_{c3,m}$
$\sigma_{cnt > 5}$
$\gamma_{c3,\ count(*) \to cnt,\ max(\ abs(p3-p4)\ ) \to m}$
$\bowtie_{oid2=oid4\ \wedge\ c3=c4}$
$\bowtie_{oid=oid3}$
$\bowtie_{pid=rid2\ \wedge\ pid2=rid}$
$\rho_{pid2,rid2,oid2}$
$\rho_{oid3,c3,p3}$
$\rho_{oid4,c4,p4}$
Gift
Gift
Item
Item

`Item(oid,category,price)`
`Gift(pid, rid, oid)`

(b) (10 points) Write a Datalog query that returns the identifier of all people who offered or received a ring as a gift but never offered nor received a book.

Solution:

RingPeople(x) :- Gift(x,_,o), Item(o,'ring',_)
RingPeople(x) :- Gift(_,x,o), Item(o,'ring',_)
BookPeople(x) :- Gift(x,_,o), Item(o,'book',_)
BookPeople(x) :- Gift(_,x,o), Item(o,'book',_)
Answer(x) :- RingPeople(x), NOT BookPeople(x)

```
Item(oid,category,price)
Gift(pid, rid, oid)
Person(pid)
```

(c) (10 points) State what each of the following relational calculus queries computes.

(a)

$$P(x) = \exists o.\exists c.\exists p.\exists r.\,(\texttt{Person(x)} \land \texttt{Item}(o, c, p) \land \texttt{Gift}(x, r, o))$$

(b)

$$P(x) = \texttt{Person(x)} \land (\forall o.\forall c.\forall p.\,(\texttt{Item}(o, c, p) \Rightarrow \exists r.\texttt{Gift}(x, r, o)))$$

(c)

$$P(x) = \texttt{Person(x)} \land (\forall o.\forall r.\,(\texttt{Gift}(x, r, o) \Rightarrow \exists p.\texttt{Gift}(p, x, o)))$$

Solution:

1. People who offered an item as a gift.
2. People who offered every single item as a gift.
3. People who offered as a gift only items they received as gift.

3 Query Evaluation

3. (10 points)

   (a) (10 points) Describe how a relational database management system (DBMS) evaluates a query. (A) What happens from the moment the DBMS receives a SQL statement in the form of a string to the moment it returns results to the user? More complete answers will receive more points. (B) How does the presence of an index affect what happens?

Solution: When a DBMS receives a query string as input, it performs the following actions:

1. Parses the string into an internal format. Checks that the query is syntactically valid, that the table names and attributes names exist, and that the user is authorized to access the given tables.
2. The query is then transformed into a logical query plan and subsequently into a physical query plan. The logical query plan is a relational algebra expression. The physical query plan is a logical query plan with extra information on how to access data from disk (e.g., use an index or not), which algorithm to use for each operator (e.g., use a hash join or a nested loop join), and whether to pipeline results between operators or materialize any intermediate relations to disk. The process of selecting the physical query plan for a query is called query optimization. In a cost-based optimizer, the DBMS selects the physical query plan with the lowest estimated cost. The cost is a function of the estimated number of I/O operations, CPU operations, and network bandwidth when applicable.
3. Finally, the selected query plan is executed by calling open()/next()/close() recursively from the top of the query plan. Results are returned to the application.

The presence of an index gives rise to a larger space of possible physical plans to consider. An index can serve to fetch data from disk or it can serve in an index-nested loop join to look up matching tuples.