CSE 344 Final Exam

March 17, 2015

Name: ______________________________

Question 1	/ 10
Question 2	/ 30
Question 3	/ 18
Question 4	/ 24
Question 5	/ 21
Question 6	/ 32
Question 7	/ 35
Question 8	/ 20
Total	/ 190

The exam is closed everything except for 2 letter-size sheets of notes. No books, computers, electronics devices, phones of the smart or not-so-smart variety, telegraphs, telepathy, tattoos, mirrors, smoke signals, or other contraptions permitted. By putting your name on this exam, you are certifying that you did not give or receive any unpermitted aid in the exam.

The exam lasts 110 min. Please budget your time so you get to all questions.

Please wait to turn the page until everyone has their exam and you are told to begin.

Relax. You are here to learn.

Reference Information

This information may be useful during the exam. Feel free to use it or not as you wish. You can remove this page from the exam if that is convenient.

Reference for SQL Syntax

Outer Joins

```
-- left outer join with two selections:
SELECT *
FROM R LEFT OUTER JOIN S on R.x=55 and R.y=S.z and S.u=99
```

The UNION Operation

```
SELECT R.k FROM R UNION SELECT S.k FROM S
```

The CASE Statement

```
SELECT R.name, (CASE WHEN R.rating=1 THEN 'like it'
                     WHEN R.rating IS NULL THEN 'do not know'
                     ELSE 'unknown' END) AS a_rating
FROM R;
```

The WITH Statement

Note: with is not supported in sqlite, but it is supported SQL Server and in postgres.

```
WITH T AS (SELECT * FROM R WHERE R.K>10)
SELECT * FROM T WHERE T.K<20
```

Reference for Relational Algebra

Name	Symbol
Selection	σ
Projection	π
Natural Join	$\bowtie$
Group By	γ
Set Difference	$-$
Duplicate Elimination	δ
Renaming of R to new relation with attributes A_1, A_2, A_3	$\rho_{A1,A2,A3}(R)$

XQuery example (from lecture slides) (a reminder of XQuery syntax)

```
FOR $b in doc("bib.xml")/bib
LET $a := avg($b/book/price/text())
FOR $x in $b/book
WHERE $x/price/text() > $a
RETURN $x
```

Question 1. (10 points, 1 point each) Warm up – True or false (circle one).

T F Broadcast join requires data to be redistributed using a hash function.

T F A serializable schedule is always conflict-serializable.

T F Two-phase locking is used to handle transactions that span multiple partitions.

T F We need locks to ensure all transactions execute serially.

T F Hash indexes benefit range selection queries.

T F A relation can have at most one unique key.

T F All XQuery outputs are well-formed XML.

T F SQL queries and relational algebra expressions are one-to-one mappings.

T F Every key is a superkey.

T F A given schema with a set of functional dependencies can have multiple minimal superkeys.

Question 2. (30 points) Return of the Dawgs.[1]

The International Sled Dog (Husky) Racing Association (ISDRA) has turned tech-savvy after the midterm! This time they decided to store their sled race information using XML with the following DTD:

```
<!DOCTYPE races [
  <!ELEMENT races (race)+>
  <!ELEMENT race (id, (participant)+)> // id uniquely identifies each race
  <!ATTLIST race date CDATA #REQUIRED> // MM/DD/YYYY
  <!ATTLIST race location CDATA #REQUIRED> // maximum 30 characters long
  <!ELEMENT participant (dog, musher)>
  <!ATTLIST participant resultPosition CDATA #REQUIRED> // = 1 if winner
  <!ELEMENT dog (id, name, age)>       // id uniquely identifies each dog
  <!ELEMENT musher (id, name)>         // id uniquely identifies each musher
  <!ELEMENT id (#PCDATA)>              // integers
  <!ELEMENT name (#PCDATA)>             // maximum 30 characters long
  <!ELEMENT age (#PCDATA)>             // integers
]>
```

Write XQuery expressions for the following queries. The data is stored on a file called `races.xml`.

a) Find the names of all the dogs that participated in races that took place at Iditarod on February 1, 2015. (7 points)

b) Find the average age of the dogs that won at least one race in Fairbanks. (7 points)

[1] Also the actual name of the Huskies 2005 Football team yearbook.

c) Convert the DTD above into a relational schema. (4 points)

d) Define a virtual view on top of your schema from c) that stores the number of distinct dogs that have raced at each location. The output schema should be `raceStats(location varchar(20), numDogsRaced int)`. (5 points)

e) Write a non-recursive Datalog query for a) using your relational schema from d). (7 points)

Question 3. (18 points) Registering for races.

The ISDRA maintains a website for racers to register for races.

Races are stored with schema: `races(mid int, did int, raceNum int)`

The following pseudo-code is used for online registration:

```
register (musherId, dogId, raceNumber):
L1: musherCount = execute(SELECT COUNT(*) FROM races WHERE raceNum = raceNumber);
L2: if (musherCount < 10)   // 10 mushers maximum per race
L3:   execute(INSERT INTO races VALUES (musherId, dogId, raceNumber));
```

a) Three different mushers attempt to register for race #5, which has only one slot left, by calling `register` from their browsers independently:

```
C1: register(1, 2, 5);
C2: register(2, 6, 5);
C3: register(3, 7, 5);
```

At the end of the day, all three of them succeeded in registering for the race! How could this happen? Show a schedule of the above commands that could result in this outcome. Indicate your answer using the labels above and assume each of `L1`, `L2`, and `L3` is executed atomically.

For instance, the schedule `C1:L1; C1:L2; C1:L3; C2:L1;` means execute `L1` from `C1`, then `L2` from `C1`, then `L3` from `C1`, etc. (6 points)

b) ISDRA realizes the error above was caused by not having any locking protocol in their DBMS. ISDRA now implements strict two-phase locking with record-level shared and exclusive locks in their DBMS, and puts `BEGIN TRANSACTION` before `L1` and `COMMIT` after `L3`. Explain why that fixes the problem in a). (6 points)

c) In addition to online registrations, the ISDRA system, now running on the DBMS from b) with the fix in `register`, also supports report generation about races using the following code:

```
generateReport (raceNumber):
L1: BEGIN TRANSACTION;
L2: records = execute(SELECT * FROM races WHERE raceNum = raceNumber);
L3: for (record : records) { print(record); }
L4: count = execute(SELECT COUNT(*) FROM races WHERE raceNum = raceNumber);
L5: print("total current registered mushers: " + count);
L6: COMMIT;
```

They notice that sometimes there is an error: the count does not match the number of records printed even after using transactions! Explain how that can happen and what can you do to fix this problem. (6 points)

Question 4. (24 points) Of Pigs and Dawgs.

ISDRA stores pedigree history of dogs using files and would like to process them using Pig. Suppose they have two Pig tables defined as follows:

```
// each rid is unique
pedigreeRecords = load ‘records.dat’ using TextLoader as (rid:int, rname:chararray);
// each pid is unique
people = load ‘people.dat’ using TextLoader as (pid:int, pname:chararray);
// each (rid, pid) pair is unique
dogOwners = load ‘owners.dat’ using TextLoader as (do_rid:int, do_pid:int);
```

Consider the following Pig program:

```
x = group dogOwners by do_rid;
x2 = foreach x generate flatten(dogOwners), COUNT(dogOwners) as count;
x3 = cogroup x2 by do_rid, pedigreeRecords by rid;
x4 = foreach x3 generate flatten(pedigreeRecords), flatten(x2);
x5 = foreach x4 generate rname, count;
dump x5; // prints result set x5
```

a) Pig implements the program above using MapReduce. Assume that each map function can read from at most one base table. How many map-reduce jobs will this program generate (hint: >1)? (3 points)

b) Implement the first `map` function (pseudocode is fine as long as you clearly state what the inputs are and what key-value pairs are generated). (7 points)

Code repeated here for your convenience.

```
// each rid is unique
pedigreeRecords = load ‘records.dat’ using TextLoader as (rid:int, rname:chararray);
// each pid is unique
people = load ‘people.dat’ using TextLoader as (pid:int, pname:chararray);
// each (rid, pid) pair is unique
dogOwners = load ‘owners.dat’ using TextLoader as (do_rid:int, do_pid:int);

x = group dogOwners by do_rid;
x2 = foreach x generate flatten(dogOwners), COUNT(dogOwners) as count;
x3 = cogroup x2 by do_rid, pedigreeRecords by rid;
x4 = foreach x3 generate flatten(pedigreeRecords), flatten(x2);
x5 = foreach x4 generate rname, count;
dump x5; // prints result set x5
```

c) Implement the first `reduce` function given the `map` function you wrote above (pseudocode is fine as long as you clearly state what the inputs are and what outputs are generated). (7 points)

d) To check query performance, the ISDRA also stores their records in the following relations:

```
pedigreeRecords(rid int, rname varchar(20))
people(pid int, pname varchar(20))
dogOwners(do_rid int, do_pid int)
```

Using these relations, show how you would rewrite the Pig program above using SQL. (7 points)

Question 5. (21 Points) Running in parallel.

The ISDRA now wants to compare PIG and parallel DBMS performance. With data stored in the following relations:

```
races(mid int, did int, raceNum int) -- stores race records
dogs(did int, name varchar(20), age int)
mushers(mid int, name varchar(20), age int)
```

They want to measure system performance with the following query:

```
SELECT d.did, COUNT(*)
FROM races r, dogs d, mushers m
WHERE r.mid = m.mid AND r.did = d.did AND m.age > 21
GROUP BY d.did
```

a) Briefly describe what this query is computing. (3 points)

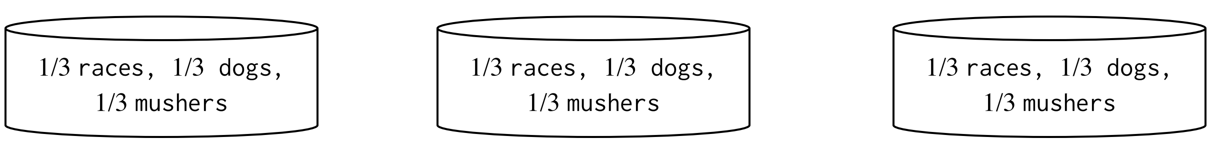
b) If you can create two indexes on the three tables to speed up the query above, what would you choose? Briefly justify your answer. (6 points)

Code repeated here for your convenience.

```
races(mid int, did int, raceNum int) -- stores race records
dogs(did int, name varchar(20), age int)
mushers(mid int, name varchar(20), age int)

SELECT d.did, COUNT(*)
FROM races r, dogs d, mushers m
WHERE r.mid = m.mid AND r.did = d.did AND m.age > 21
GROUP BY d.did
```

c) Suppose `races`, `dogs`, and `mushers` are block-partitioned across three different machines. Draw out how the query will be executed by a parallel DBMS that implements all joins using shuffle (repartition) joins assuming no indexes are available. Clearly label what each step is performing. (7 points)



f) Briefly describe what happens to the query plan above if the data is hash-partitioned rather than block-partitioned. (5 points)

Question 6. (32 points) The bookstore.

After the midterm, the ISDRA bookstore is now under new management! For starters, they would like to redesign their DBMS.

a) Design an E/R diagram for the bookstore that contains the following objects and their attributes: (10 points)

- periodicals: name, issue number, publisher
- fiction: name, author, publisher
- catalogs: name, issue number, publisher
- stores: zip code, square footage
- newsstands: zip code

Model the following relationships among the objects:

- Each periodical contains review of at most one other fiction or periodical.
- Each catalog contains an advertisement of at most one other catalog, fiction, or periodical.
- Stores sell only fiction.
- Newsstands sell only periodicals and catalogs

b) Write the `CREATE TABLE` statements to represent this E/R diagram using SQL relations. Clearly label all keys and foreign keys. (10 points)

Learning from HW4, the store maintains the following relations for its employee records:

```
employee (officeNum, SSN, phone, managerName, deptNum)
```

Given the following functional dependencies:

```
officeNum → phone
SSN → officeNum, deptNum
deptNum → managerName
```

c) List one key of the employee relation. (5 points)

d) Is the employee relation in BCNF? If so, write "Yes" below. Otherwise, decompose it into BCNF and underline all keys and foreign keys in the final relations. (7 points)

Question 7. (35 points) Trouble at the plant.

The bookstore has been running different transactions on its inventory table with schema:

```
inventory (bid int, price double, count int) -- attributes abbreviated as (b, p, c)
```

and using the following transactions:

```
T1: R(b); R(p); R(c); W(c);
T2: R(b); W(c); R(c);
T3: R(b); W(p); R(c); W(p);
```

For each of the schedules shown in a) to d), circle all categories that the given schedule satisfies. (4 points each)

a) `R1(b); R2(b); R1(p); R1(c); R3(b); W1(c); W2(c); W3(p); R3(c); R2(c); W3(p);`

Serial Serializable Conflict-serializable Not serializable

b) `R3(b); R1(b); R1(c); W3(p); R2(b); R1(p); W2(c); W3(p); R2(c); R3(c); W1(c);`

Serial Serializable Conflict-serializable Not serializable

c) `R2(b); R1(b); R3(b); W3(p); W2(c); R1(p); R1(c); R3(c); W1(c); W3(p); R2(c);`

Serial Serializable Conflict-serializable Not serializable

Code repeated here for convenience.

```
T1: R(b); R(p); R(c); W(c);
T2: R(b); W(c); R(c);
T3: R(b); W(p); R(c); W(p);
```

d) Under what isolation level is the following schedule allowed?

```
R3(b); R1(b); W3(p); R2(b); R1(p); R1(c); W2(c); W1(c); R3(c); R2(c); W3(p);
```

Read uncommitted Read committed Repeatable read Serializable

e) Draw the precedence graph for the schedule shown in d). (7 points)

Code repeated here for your convenience.

```
T1: R(b); R(p); R(c); W(c);
T2: R(b); W(c); R(c);
T3: R(b); W(p); R(c); W(p);
```

Consider this schedule:

```
R2(b); R3(b); W3(p); W2(c); R3(c); R3(c); W3(p);
```

f) Could it be produced by a scheduler using two-phase locking with only exclusive locks? If yes, show the schedule with locking operations (Use `L1(b)` to indicate `T1` locking on attribute `b`, and `U1(b)` to indicate `T1` unlocking attribute `b`). If no, briefly explain why not. (4 points)

g) Could it be produced by a scheduler using two-phase locking with shared and exclusive locks? If yes, show the schedule with locking operations. If no, briefly explain why not. (4 points)

h) Finally, could it be produced by a scheduler using strict two-phase locking with shared and exclusive locks? If yes, show the schedule with locking operations. If no, briefly explain why not. (4 points)

Question 8. (20 points, 5 points each) Short Answers.

a) What is the difference between horizontal and vertical partitioning?

b) When would you use a virtual view as opposed to a materialized view and why?

c) List out what ACID stands for and explain two of them.

d) List one data model that is used in NoSQL systems other than relations.

END OF EXAM

Thank you for making the class enjoyable! Hope you have learned tons.
Good luck with finals and have an awesome spring break!
– 344 staff –