Endterm Exam *(Version A)*
CS 122A
Spring 2017

Max. Points: 100
(please read the instructions carefully)

Instructions:
- The total time for the exam is 120 minutes; be sure to *budget your time* accordingly.
- The exam is closed book and closed notes but “open cheat sheet”.
- Read each question first, in its entirety, and then carefully answer each part of the question.
- If you don’t understand something, ask one of the exam proctors for clarification.
- If you still find ambiguities in a question, note the interpretation you are taking.

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>TOPIC</th>
<th>POINTS</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fake News?</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>Index Structures</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>SQL Queries</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>Short Answers</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Physical Design</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>NoSQL</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>TOTAL</td>
<td>All</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Question 1: Fake News? (20 points)

(2 pts each) For each of the following statements issued by Sean Spicer, indicate whether the given statement is TRUE (circle TRUE) or FALSE (circle FALSE):

Aggregates in SQL give it more expressive power than the relational calculus.

TRUE          FALSE

The performance of an index-only query plan will be better if the relevant index is clustered.

TRUE          FALSE

Locking is the mechanism used in database systems to ensure that a transaction’s result is durable.

TRUE          FALSE

Check constraints in SQL cannot involve joins.

TRUE          FALSE

OUTER joins are only allowed at the top-level of a SQL query that involves nesting (hence the name).

TRUE          FALSE

B+ trees built by an initial load are faster to construct, but then slower to query, than the “same” B+ tree (a B+ tree with the same data content) would be if it were built using repeated insertions.

TRUE          FALSE

In a static hashed index, a tree search is used to locate the hash bucket (and overflow pages) that might hold the record with a given key value.

TRUE          FALSE

The provision of schema information is optional in NoSQL databases.

TRUE          FALSE

All NoSQL databases are simple key-value stores without declarative query support, hence the name.

TRUE          FALSE

Triggers are more powerful than both CHECK constraints and FOREIGN KEY constraints.

TRUE          FALSE
Question 2: Index Structures (15 points)

Consider the following example of a B+ tree index of order d=2. Suppose that this is a secondary index on Users.age with leaf entries that contain key/record-id pairs.

![B+ Tree Diagram](image)

(6 pts) Draw the B+ tree structure that would result from doing the following SQL operation:

\[
\text{INSERT INTO Users (id, name, phone_num, age) VALUES (123, 'Iam A. Kid', '949-555-1212', 20);}
\]

(6 pts) Draw the index that would result from doing the following SQL operation with the original index:

\[
\text{DELETE FROM Users WHERE age = 47;}
\]

(3 pts) How many index page reads would be needed to run the following query with the original index? Justify your answer by listing the affected page IDs:

\[
\text{SELECT * FROM Users WHERE age BETWEEN 40 AND 49;}
\]

# reads: 

\[
\text{page IDs: P1, P4, P5, P6}
\]
Question 3: SQL Queries (30 pts)

Consider the following relational database schema – along with the sample data on the next page – for keeping track of UCI students and their major departments. The primary key for the Student table is sid, the primary key for the Dept table is dno, and there are two foreign keys in Student: one is major, which references Dept.dno, and the other is mentor, which references Student.sid. (A student can have another student assigned to help mentor them throughout the course of their studies.)

Tables: **Student**(sid, sname, age, year, major, mentor)  **Dept**(dno, dname, school, chair)

Answer each of the following questions using SQL. (Avoid duplicate rows in your query results.)

(6 pts) Print the school and chair of departments with at least one major who is less than 18 years old.

```
SELECT D.school, D.chair
FROM Dept D
WHERE EXISTS (SELECT * FROM Student S
    WHERE S.major = D.dno AND S.age < 18);
```

(6 pts) Print the school and chair of the department(s) that has the oldest student majoring in it.

```
SELECT DISTINCT D.school, D.chair
FROM Student S
INNER JOIN Dept D ON (S.major = D.dno)
WHERE S.age = (SELECT MAX(age) FROM Student);
```

(6 pts) Restate the purpose of the following SQL query in simple English.

```
SELECT dname FROM Dept D
WHERE NOT EXISTS
    (SELECT * FROM Student S
    WHERE S.major = D.dno AND NOT (S.year = ‘Grad’));
```

Print the names of the departments that only have graduate students.
Question 3: SQL Queries (continued)

Here again is the relational schema – and also some sample data – for keeping track of UCI students and their major departments. The primary key for Student is sid, the primary key for Dept is dno, and there are two foreign keys in Student: major, referencing Dept.dno, and mentor, referencing Student.sid.

Tables: Student(sid, sname, age, year, major, mentor)  Dept(dno, dname, school, chair)

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>age</th>
<th>year</th>
<th>major</th>
<th>mentor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sally</td>
<td>22</td>
<td>Junior</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>18</td>
<td>Freshman</td>
<td>null</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Rahul</td>
<td>22</td>
<td>Grad</td>
<td>30</td>
<td>null</td>
</tr>
<tr>
<td>4</td>
<td>Jake</td>
<td>null</td>
<td>Senior</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Leo</td>
<td>33</td>
<td>Grad</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>Emily</td>
<td>14</td>
<td>Freshman</td>
<td>50</td>
<td>null</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

(6 pts) Show what the result would be from executing the following SQL query on the sample data above.

```
SELECT S.sid, S.sname, S.age, M.sname AS mentor_name, M.age AS mentor_age
FROM (Student S JOIN Dept D ON (S.major = D.dno))
LEFT OUTER JOIN Student M ON (S.mentor = M.sid)
WHERE D.school = 'PhySci';
```

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>age</th>
<th>mentor_name</th>
<th>mentor_age</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Rahul</td>
<td>22</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>5</td>
<td>Leo</td>
<td>33</td>
<td>Rahul</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>Emily</td>
<td>14</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>

(6 pts) Create a relational view SchoolStats(school, num_students, min_age, max_age) that can be used by the university’s deans to query the student demographic information for the university’s schools. Be sure to include all schools, even those with no students.

```
CREATE VIEW SchoolStats(school, num_students, min_age, max_age) AS
SELECT D.school, COUNT(S.sid), MIN(S.age), MAX(S.age)
FROM Dept D LEFT OUTER JOIN Student S ON S.major = D.dno
GROUP BY D.school;
```
Question 4: Short Answers (10 points)

(2 pts) Suppose that three transactions, T1, T2, and T3, are submitted to a transactional DBMS at the same time and that the DBMS is set up to run at the highest (SERIALIZABLE) SQL consistency level. Which of the following possible serial-equivalent execution orders might end up occurring and be considered as being correct outcomes? (Circle all that apply.)

- T1, T2, T3
- T2, T1, T3
- T1, T3, T2
- T3, T1, T2
- T3, T2, T1
- T2, T3, T1

(2 pts) When drawing pictures of indexes in class, we have been careful to put an asterisk next to the search key values shown in their leaf pages (e.g., 60* rather than simply 60 for the key value 60). Why? (Be brief!)

We do this to remind ourselves that the leaves hold (k, i(k)) pairs, not just k’s.

(2 pts) Which of the following is not a valid JSON object? (Circle the one that isn’t.)

a. {"sid":39, "sname": "James", "schools": ["UCI", "UCLA", "UCSD", {"degree": "bachelor"}, 2017]}

b. {"user": {"id": 39, "addr": "Irvine, CA", "zip":92617, "graduated":false, "phone":"949-123-4568"}}

c. {"menuitem": [1,5,9], "reservation_date": "2016-10-17", "isConfirmed":true, "amount":"$100"}

(4 pts) A SQL database can be made more secure by combining its access control capabilities (i.e., GRANT / REVOKE) with which of the following other SQL DDL features? (Check all that apply.)

- Indexes
- Views
- Foreign keys
- Stored procedures
- Check constraints
- Unique constraints
Question 5: Physical Design (10 points)

Here one last time is the relational schema for keeping track of UCI students and their major departments. The primary key for Student is sid, the primary key for Dept is dno, and the two foreign keys in Student are major, referencing Dept.dno, and mentor, referencing Student.sid.

Tables: Student(sid, sname, age, year, major, mentor)  Dept(dno, dname, school, chair)

The time has come to design a good physical schema to sit underneath this logical schema. Your job is thus to design a physical storage and indexing strategy to serve the following expected query workload well. The two primary keys have already been indexed, with unclustered indexes, and we have started filling in the table below with those. Assume that the university is using an open source DBMS that has support for B+ tree indexes of both the clustered and unclustered variety. Finish filling in the table to show all of your indexing recommendations for the query mix below; feel free to add additional motivations to the given rows if appropriate.

Q1: SELECT COUNT(*) FROM Student S WHERE S.age > value1;
Q2: SELECT * FROM Dept D WHERE D.school = value1;
Q3: SELECT * FROM Student S JOIN Dept D ON (S.major = D.dno) WHERE S.sid = value1;
Q4: SELECT * FROM Student S WHERE S.year LIKE ‘%value1’;
Q5: SELECT S.sname, COUNT(*) AS scnt FROM Student S GROUP BY S.sname ORDER BY scnt;

<table>
<thead>
<tr>
<th>Motivation(s)</th>
<th>Indexed Table</th>
<th>Indexed Column(s)</th>
<th>Clustered or Unclustered?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Key, Q3</td>
<td>Student</td>
<td>sid</td>
<td>Unclustered</td>
</tr>
<tr>
<td>Primary Key, Q3</td>
<td>Dept</td>
<td>dno</td>
<td>Unclustered</td>
</tr>
<tr>
<td>Q1</td>
<td>Student</td>
<td>age</td>
<td>Unclustered</td>
</tr>
<tr>
<td>Q2</td>
<td>Dept</td>
<td>school</td>
<td>Clustered</td>
</tr>
<tr>
<td>Q5</td>
<td>Student</td>
<td>sname</td>
<td>Unclustered</td>
</tr>
</tbody>
</table>
**Question 6: NoSQL (15 points)**

Let’s make one more use this term of our favorite database from class - **Sailors, Boats, and Reserves**. The following tables show portions of some actual instances of the three tables on a MySQL database.

<table>
<thead>
<tr>
<th>Sailors</th>
<th>Reserves</th>
<th>Boats</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>sname</td>
<td>rating</td>
</tr>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
</tr>
<tr>
<td>64</td>
<td>Horatio</td>
<td>7</td>
</tr>
<tr>
<td>71</td>
<td>Zorba</td>
<td>10</td>
</tr>
<tr>
<td>74</td>
<td>Horatio</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A DBA has decided to migrate these tables into an AsterixDB instance. The following AsterixDB DDL shows the DBA’s plan for organizing the sailing club data in this new non-1NF world:

```sql
CREATE TYPE RType AS {
    bid: int,
    rdate: date
}

CREATE TYPE BType AS {
    bid: int,
    bname: string,
    color: string?
}

CREATE TYPE SType AS {
    sid: int,
    sname: string,
    rating: int?,
    age: int?,
    reserv: [ RType ]
}

CREATE DATASET Sailors(SType)
    PRIMARY KEY sid;

CREATE DATASET Boats(BType)
    PRIMARY KEY bid;
```

(3 pts) Briefly describe one advantage and one disadvantage of the above schema design for AsterixDB as compared to the original relational schema:

**Advantage:** Fewer joins required; ....

**Disadvantage:** Sailors with many reservations can be big; making a reservation means updating a sailor; ....
(4 pts) Since designing the AsterixDB schema, the DBA has learned more about the sailing club. She now knows that the typical sailor has about five reservations at a time and that a typical boat can easily have a few hundred reservations at a time. If she wants to keep a design that doesn’t use a separate Reservations dataset, should she revise the above schema? Circle your answer (Yes or No) below and briefly describe your reasoning.

Should she revise the schema?   Yes   No

Reason: We want to keep the record sizes reasonable. As a result, it is better to have reservation information in Sailors as opposed to having reservation information in Boats since the resulting record sizes will be smaller and hence more manageable to query and update.

(4 pts) The DBA needs to tutor the application developers about NoSQL data and AsterixDB. She wants to show them a sample NoSQL data instance in the (JSON-like) Asterix Data Model. Finish filling in the following sailor object’s contents based on the original AsterixDB schema and the sample relational data.

```json
{
  "sid" : 31,
  "sname" : "Lubber",
  "rating" : 8,
  "reserv" : [{
    "bid" : 102,
    "rdate" : date("2016-11-10")
  },
  {"bid" : 103,
    "rdate" : date("2016-11-06")}
}
```

(4 pts) The DBA has successfully generated the new database (a dataverse!) using the original AsterixDB DDL and populated the datasets. She’s written several queries and wants to share them with the DBA who manages the MySQL database. Write an equivalent SQL query (for MySQL) for the following SQL++ query that she wrote for AsterixDB:

```sql
SELECT bid, COUNT(*) AS cnt
FROM (SELECT R.bid
       FROM Sailors S
       UNNEST S.reserv R
       WHERE R.rdate > date("2016-10-01")
     ) S2
GROUP BY S2.bid
HAVING cnt >= 2
```

SELECT bid, COUNT(*) AS cnt
FROM Reserves
WHERE rdate > "2016-10-01"
GROUP BY bid
HAVING cnt >= 2