Endterm Exam (Version A)
CS 122A
Winter 2017
Max. Points: 100
(Please read the instructions carefully)

Instructions:
- The total time for the exam is up to 90 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>Indexing</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Physical Database Design</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NoSQL Databases</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Transactions</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>All</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Question 1: Indexing (30 points)

(a) (5 pts) The leaf entries in each of the index structures that we have studied are pairs (k, I(k)), or in the book’s terms, (k, k*), where k is a key value and I(k) (or k*) is its associated information (a data entry). Suppose that we have a table Emp(eno, name, age, salary, deptno) with a primary index defined on the table’s key column Emp.eno and a secondary index on the column Emp.age.

(i) Which of the following are sensible options for what I(k) could be, in general, for the primary Emp.eno index? Indicate your choices by marking their boxes with an X.

- I(k) = the record id for a unique Emp record with the associated Emp.eno value
- I(k) = one or more record ids for the Emp records with the associated Emp.eno value
- I(k) = the primary key for a unique Emp record with the associated Emp.eno value
- I(k) = one or more primary keys for the Emp records with the associated Emp.eno value
- I(k) = the unique Emp record with the associated Emp.eno value
- I(k) = one or more Emp records with the associated Emp.eno value

(ii) Which of the following are sensible options for what I(k) could be, in general, for the secondary Emp.age index? Indicate your choices by marking their boxes with an X.

- I(k) = the record id for a unique Emp record with the associated Emp.age value
- I(k) = one or more record ids for the Emp records with the associated Emp.age value
- I(k) = the primary key for a unique Emp record with the associated Emp.age value
- I(k) = one or more primary keys for the Emp records with the associated Emp.age value
- I(k) = the unique Emp record with the associated Emp.age value
- I(k) = one or more Emp records with the associated Emp.age value

(b) (10 pts) We’ve studied Static Hashed indexes, ISAM indexes, and B+ Tree indexes. Below are some properties of one or more of these structures. For each one, circle the structure(s) that it is a property of.

(i) Supports efficient (compared to a file scan) exact-match key lookup queries

- Static Hashed
- ISAM
- B+ Tree

(ii) Supports efficient key range queries

- Static Hashed
- ISAM
- B+ Tree

(iii) Ensures equal I/O cost for every exact-match key lookup (in terms of number of pages read)

- Static Hashed
- ISAM
- B+ Tree

(iv) Employs overflow chaining in order to handle page capacity overflows

- Static Hashed
- ISAM
- B+ Tree

(v) Has the possibility that a key insertion may cause its height to increase by one level

- Static Hashed
- ISAM
- B+ Tree

(vi) Often performs an exact-match lookup, even for a very large file, with just one disk read

- Static Hashed
- ISAM
- B+ Tree

SCORE: _________
Question 1: Indexing (cont’d.)

Consider the following sketch of a B+ Tree index on the Person table’s field Person.age. Note that the index pages are labeled for reference in the questions that follow.

(c) (5 pts) Assuming that this index is on disk, list (in order!) the index pages that must be read from disk in order to execute the query `SELECT * FROM Person WHERE age > 40 AND age <= 65`:

(d) (5 pts) Show what the above index will look like following the insertion of a new Person record for someone on their 40th birthday.

(c) (5 pts) Show what the original index from above will look like following the untimely death (and then deletion) of the 93 year old in the Person table.

SCORE: _________
Question 2: Transactions (20 points)

Consider your favorite table, which is Sailors of course, from this term’s lectures:

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bob</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>Sally</td>
<td>9</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>Zack</td>
<td>10</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>Abby</td>
<td>3</td>
<td>null</td>
</tr>
<tr>
<td>5</td>
<td>Joe</td>
<td>null</td>
<td>35</td>
</tr>
</tbody>
</table>

Now consider the following SQL transactions that users might wish to run against this table:

**Transaction T1:**

```
BEGIN TRANSACTION;
    UPDATE Sailors SET rating = rating * 2
    WHERE sname = 'Abby';
    DELETE FROM Sailors WHERE sname = 'Bob';
END TRANSACTION;
```

**Transaction T2:**

```
BEGIN TRANSACTION
    SELECT sname FROM Sailors WHERE rating > 4;
END TRANSACTION;
```

(a) (5 pts) Suppose you are attempting to run transaction T1 on your laptop, but you accidentally drop your laptop (!) just after hitting ‘Enter’ to submit the transaction to your favorite database system. After verifying that your keyboard and disk survived the accident, you let the database system run recovery and then you run transaction T2. Considering all the possibilities in terms of how far along T1 got before your laptop hit the floor, list all possible answers from T2 and briefly explain how they could occur.

(b) (5 pts) Now suppose that you were to run transactions T1 and T2 concurrently in different windows on your laptop. Further suppose that you have wisely configured your database system’s consistency level to **SERIALIZABLE**. Again list all possible answers from T2 and briefly explain them.
Question 2: Transactions (cont’d.)

(c) (10 pts) Circle the correct answer(s) to each of the following short questions about transactions.

(i) This ACID property ensures that once a user learns that their transaction has committed successfully, they don’t have to worry about its data being lost.

Atomicity  Consistency  Isolation  Durability

(ii) This ACID property relieves application builders from having to worry about potential interactions between their application program and other developers’ programs.

Atomicity  Consistency  Isolation  Durability

(iii) This ACID property is something that an application builder has to be at least partly concerned with when writing their application code in order for the other ACID properties to ensure correctness despite other things that might go wrong when running in a multi-user and potentially failure-prone environment.

Atomicity  Consistency  Isolation  Durability

(vi) In two-phase locking (2PL), transactions automatically acquire shared (S) locks on objects that they read and exclusive (X) locks on objects that they write.

TRUE        FALSE

(v) Suppose that 5 concurrent transactions are concurrently attempting to read the same record in a table holding bank account information. How many of these transactions will be able to read the record at one time under 2PL?

0 1 2 3 4 5

(vi) Suppose that 5 concurrent transactions are concurrently attempting to update the same record in a table holding bank account information. How many of these transactions will be able to write the record at one time under 2PL?

0 1 2 3 4 5

(vii) Depending on what they are doing, it is possible for concurrent transactions to become involved in a deadlock under strict two-phase locking. If this happens, it can lead to an inconsistent database state.

TRUE        FALSE

(viii) In write-ahead logging, transactions automatically record which of the following information in the system’s transaction log in order to be able to identify and either rollback or replay lost transactions in the event of various potential failures during or after their execution? (Indicate each one that applies with an X in the box.)

- The before-images of the records that they read
- The before-images of the records that they update
- The after-images of the records that they update
- Their final transactional outcome (COMMIT or ABORT)

SCORE: _________
Question 3: Physical Database Design (25 points)

Consider again our favorite database:

Sailors(sid, sname, rating, age)
Reserves(sid, bid, date)
Boats(bid, bname, color)

You may assume that unique, unclustered B+ Tree indexes have already been created by the system on the indicated primary keys of the Sailors and Boats tables. Consider a physical design for supporting a workload that consists of the following expected query mix:

Q1: SELECT * FROM Boats WHERE bname LIKE '%value1%';
Q2: SELECT color, COUNT(*) FROM Boats GROUP BY color;
Q3: SELECT B.bname, B.color FROM Boats B, Reserves R, Sailors S
    WHERE S.age >= value1 AND S.sid = R.sid AND R.bid = B.bid;
Q4: SELECT * FROM Sailors WHERE sname LIKE 'value1%'
Q5: SELECT age, AVG(rating) FROM Sailors GROUP BY age;

(a) (10 pts) Suppose you are managing this data using a DBMS that has only B+ Trees. Choose a good set of indexes for the workload above – indicating, for each index, the indexed columns and which query or queries above led you to choose that index. Use the first column in the table to number the indexes from 1 to N for quick reference in subsequent parts of this problem. (We’ve entered the first two for you.)

<table>
<thead>
<tr>
<th>Index Number</th>
<th>Indexed Table</th>
<th>Indexed Column(s)</th>
<th>Motivating Query(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sailors</td>
<td>sid</td>
<td>(System-created: PK)</td>
</tr>
<tr>
<td>2</td>
<td>Boats</td>
<td>bid</td>
<td>(System-created: PK)</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 3: Physical Database Design (cont’d.)

(b) (6 pts) Your table ought to have at least one entry for each of the database’s tables. Indicate below (by their index numbers) which of the indexes in your table should be clustered, if any, and briefly say why.

(i) Clustered index(es) for Sailors:

(ii) Clustered index(es) for Boats:

(iii) Clustered index(es) for Reserves:

(c) (4 pts) Does your initial physical design allow query Q2 to be evaluated using an index-only query plan? Answer “yes” or “no” and then explain briefly why or why not. If it does not, propose a change that will make your answer be a “yes” instead of a “no”.

(i) Initial design is index-only: yes no

(ii) Brief explanation and re-design if needed:

(d) (5 pts) Does your initial physical design allow query Q5 to be evaluated using an index-only query plan? Answer “yes” or “no” and then explain briefly why or why not. If it does not, propose an indexing schema that will make your answer be a “yes” instead of a “no”.

(i) Initial design is index-only: yes no

(ii) Brief explanation and re-design if needed:
Question 4: NoSQL Databases (25 points)

(a) (3 pts) Which one of the following is not a valid JSON object? (Put an X by the invalid one and circle the offending portion of the data.)

☐ { "name": "Smiley", "age": 20, "phone": {}, "email": "smiley@xyz.com", "happy": true }

☐ { "name": "Smiley", "age": 20, "phone": "888-123-4567", "email": "smiley@xyz.com", "happy": yes }

☐ { "name": "Smiley", "age": "20", "phone": "888-123-4567", "email": "smiley@xyz.com", "happy": true }

☐ { "name": "Smiley", "age": 20, "phone": "888-123-4567", "email": "smiley@xyz.com", "happy": true }

(b) (3 pts) Which one of the following is not a valid JSON array? (Put an X by the invalid one and circle the offending portion of the data.)

☐ [ 1, 2, "dog", "cat", true, false, [1, "dog", null], {"pet":"dog", "fun":true} ]

☐ [ 1, 2, "dog", "cat", true, false, [1, "dog", null], {"pet":"dog", "fun":true} ]

☐ [ 1, 2, "dog", "cat", true, false, [1, "dog", null], {"dog", 13} ]

☐ [ 1, 2, "dog", "cat", true, false, [], {"pet":"dog", "fun":true} ]

(c) (4 pts) Consider the following JSON data:

```
{ "A": [1, 1, 2, 2],
  "B": {"C": 3, "D": 4},
  "E": [5, 6, true],
  "F": {"G": [null, 7]}
}
```

Which of the following could not be included as part of a JSON Schema specification that is satisfied by the JSON data above? Assume that every letter ("A", "B", "C", ...) appears in the JSON Schema specification exactly once. (Put an X by your answer and circle the offending part of the specification.)

☐ "B": {"type":"object", "additionalProperties":false "properties": {"C": {"type":"integer"}}}]

☐ "A": {"type":"array", "maxitems":10, "items":{"type":"integer"}}]

☐ "B": {"type":"object", "properties": {"C": {"type":"integer"}}}]

☐ "B": {"type":"object", "properties": {"C": {"type":["integer","null"]}, "D": {"type":["integer","null"]} }]

SCORE: _________
Question 4: NoSQL Databases (cont’d.)

Suppose that you have a dataverse containing the following datatypes and datasets in AsterixDB:

```plaintext
CREATE TYPE ChirpUserType AS {
    screenName: string,
    lang: string,
    friendsCount: int,
    name: string,
    followersCount: int?
};

CREATE TYPE ChirpMessageType AS closed {
    chirpId: string,
    user: string,
    senderLocation: point?,
    sendTime: datetime,
    referredTopics: [{ string }],
    messageText: string
};

CREATE DATASET ChirpUsers(ChirpUserType)
PRIMARY KEY screenName;

CREATE DATASET ChirpMessages(ChirpMessageType)
PRIMARY KEY chirpId;
```

And, suppose that the following SQL++ statements are run to insert data into the two datasets above:

```sql
INSERT INTO ChirpUsers ({{
    "screenName": "JoeBlow", "lang": "en", "friendsCount": 85,
    "name": "Joseph Blowhard", "followersCount": 100},
    "screenName": "SallyFifth", "lang": "en", "friendsCount": 125,
    "name": "Sally Forth"},
    "screenName": "FakeDonaldTrump", "lang": "en", "friendsCount": 6,
    "name": "Donald Trump", "followersCount": 66}
});

INSERT INTO ChirpMessages ({{
    "chirpId": "1", "user": "JoeBlow", "senderLocation": point("10.0,10.0"),
    "sendTime": datetime("2017-03-21T10:00:00"), "referredTopics": [{"databases", "sql"}],
    "messageText": "I don't want CS122a to end!"),
    "chirpId": "2", "user": "JoeBlow",
    "sendTime": datetime("2017-03-21T10:00"), "referredTopics": [{"sql", "nosql"}],
    "messageText": "I prefer SQL++ to SQL.",
    "chirpId": "3", "user": "JoeBlow", "senderLocation": point("10.0,10.0"),
    "sendTime": datetime("2017-03-21T20:00"), "referredTopics": [{"asterixdb", "fame"}],
    "messageText": "AsterixDB should be famous!"),
    "chirpId": "4", "user": "SallyFifth",
    "sendTime": datetime("2017-03-19T12:00:00"), "referredTopics": [{"life", "sql"}],
    "messageText": "Friday nights will soon be boring.",
    "chirpId": "5", "user": "FakeDonaldTrump", "senderLocation": point("0.0,0.0"),
    "sendTime": datetime("2017-03-19T12:00:00"), "referredTopics": [{"fame", "fortune", "glory"}],
    "messageText": "I have well over a million followers!"),
    "chirpId": "6", "user": "FakeDonaldTrump", "senderLocation": point("0.0,0.0"),
    "sendTime": datetime("2017-03-19T12:00:00"), "referredTopics": [{"fame", "glory"}],
    "messageText": "I have all the best people as friends, not everyone knows that; sad.",
    "chirpId": "7", "user": "FakeDonaldTrump", "senderLocation": point("0.0,0.0"),
    "sendTime": datetime("2017-03-19T12:00:00"), "referredTopics": [{"asterixdb", "glory"}],
    "messageText": "Russia clearly needs AsterixDB..."
});
```
Question 4: NoSQL Databases (cont’d.)
(d) (5 pts) Show the output of the following SQL++ query given the above data:

```
SELECT u.name AS uname,
    (SELECT VALUE m.messageText FROM ChirpMessages m
        WHERE m.user = u.screenName) AS msgs
FROM ChirpUsers u
WHERE u.screenName LIKE "Joe%";
```

(e) (4 pts) Write a brief English translation of the following SQL++ query:

```
SELECT DISTINCT m.user
FROM ChirpMessages m
WHERE (SOME t IN m.referredTopics SATISFIES t = "fame")
    OR "fortune" IN m.referredTopics;
```

(f) (6 pts) Which of the following are true in the world of AsterixDB and NoSQL database design?

(i) Instances of data objects are permitted to have extra fields that didn’t appear in their schema.

TRUE    FALSE

(ii) Multivalued (non-1NF) attributes from an ER schema require the creation of an additional dataset.

TRUE    FALSE

(iii) Composite attributes from an ER schema can be handled without flattening or the creation of an additional data set.

TRUE    FALSE