Name

CptS451 Database Systems
Spring 2017

Sakire Arslan Ay, PhD

School of Electrical Engineering and Computer Science
Washington State University

Sample Final Exam

Time Limit: 120 minutes

- Print your name and WSU ID below. In addition, initial your name in the upper right corner of every page.

Name: _____Solution key____ WSU ID:____solution key_____

- Including this cover page, this exam booklet contains 16 pages. Check if you have missing pages.
- The exam is closed book and closed notes. You are allowed to use a letter size cheat sheet (you may use both sides).
- No calculators or other electronic devices are permitted. Any form of cheating on the examination will result in a zero grade.
- Please write your solutions in the spaces provided on the exam. You may use the blank areas and backs of the exam pages for scratch work.
- Please make your answers clear and succinct; you will lose credit for verbose or confusing answers. Simplicity does count!
- The exam has 8 questions.
- You should look through the entire exam before getting started, to plan your strategy.

Q1 (12pts)	Q2 (8pts)	Q3 (16pts)	Q4 (16pts)	Q5 (20pts)	Q6 (12pts)	Q7 (6pts)	Q8 (10pts)	Total

Name

Question 1 (12pts) True/False Questions

For each of the following statements indicate whether it is TRUE or FALSE by circling your choice. **(Each question is worth 1 pts)**

(a) (1.5pts) Every relationship in an E-R diagram must translate to an individual relation in the relational model.

TRUE / FALSE

Answer: False

(b) (1.5pts) Given an SQL query, there are often multiple ways of writing it in relational algebra.

TRUE / FALSE

Answer: True

(c) (1.5pts) For any relation, it is possible to construct two separate unclustered indexes on different keys.

TRUE / FALSE

Answer: True

(d) (1.5pts) Given a relation that is not in BCNF there is always a **unique** decomposition into relations that are in BCNF.

TRUE / FALSE

Answer: False

(e) (1.5pts) B+ trees can include duplicate keys.

TRUE / FALSE

Answer: True

(f) (1.5pts) Consider the below relation:

```
Student(sid, name, major, age)
```

A clustered hash index on <major> for Student will help to run the following query efficiently.

```
SELECT sid FROM Student WHERE  major like '%EE%';
```

TRUE / FALSE

Answer: False

(a) (1.5pts) If A → BC and B→ D holds on relation R(A,B,C,D) then A → CD also holds on R.

TRUE / FALSE

Answer: True

(b) (1.5pts) Consider the below relation:

movies (name, year, director, rank)

```
Q1:  SELECT *
     FROM movies AS m1 LEFT OUTER JOIN movies AS m2
         ON m1.name=m2.name
     WHERE m1.year=2000 AND m2.rank<7.3

Q2:  SELECT *
     FROM movies WHERE year = 2000 AND rank < 7.3
```

The above queries return the same result.

TRUE / FALSE

Answer: False

Question 2 (8pts) Multiple Choice Questions

(4pts) 2.a) Which of the following statements correctly describe the difference between the HAVING and the WHERE clauses in SQL?

a) The WHERE clause tests conditions on aggregated rows resulting from the evaluation of the FROM clause; the HAVING clause tests conditions on individual rows resulting from the execution of sub-queries;
b) The WHERE clause tests conditions on individual and aggregated rows resulting from the evaluation of the FROM clause; the HAVING clause tests conditions only on aggregations calculated after the application of the GROUP BY clause;
c) The WHERE clause tests conditions on individual rows resulting from the evaluation of the FROM clause; the HAVING clause tests conditions on aggregations calculated after the application of the GROUP BY clause;
d) The WHERE clause is part of the SQL standard, and can be used in any relational DBMS systems; the HAVING clause is not part of the SQL standard, so it is not supported in all systems.

Answer: (c)

(4pts) 2.b) Consider a relation R(a,b,c) with 10,000 records, and 100 pages (100 tuples on each page). R.a is the primary key and is a non-negative integer. How many pages will be read from disk to answer the selection query $\sigma_{a<2500}(R)$ for the following scenario?

"Relation R is stored in a heap file. There also exists an unclustered B+ tree index with search key **a**. The height of the B+tree is 3. Assume that 100 records match the selection predicate."

a) 13
b) 1000
c) 1003
d) 103
e) None of the above

Answer: (d)

Name

Question 3 (16pts) SQL and Relational Algebra

The following questions refer to the database schema below:

Book(bookid, title, author, publisher, year, price)
Bookstore(name, address, city)
Sold(name, bookid, quantity)

bookid is the primary key for relation Book; name is the primary key for relation Bookstore; (name,bookid) is the primary key for relation Sold.
Sold.bookid is a FK referencing Book.bookid and Sold.name is a FK referencing Bookstore.name.

Only the first author for each book is recorded in the book table. The quantity attribute in Sold is at least 1 (i.e. quantity >=1).

a) Write the SQL SELECT query for the following:

"Find the names of the authors whose books were sold more than 10,000 copies in the city of Seattle. "

Solution:

```
SELECT author
FROM Book, Sold, Bookstore
WHERE Book.bookid =Sold.bookid
  AND Sold.name=Bookstore.name
  AND City='Seattle'
GROUP BY author
HAVING sum(quantity)>10,000;
```

b) Write the SQL SELECT query for the following:

"Find the names of the bookstores which have sold more books than any other bookstore in the same city."

Solution:

```
SELECT name
FROM Sold as S1, Bookstore as B1
WHERE S1.name =B1.name
GROUP BY S1.name
HAVING sum( S1.quantity ) >=
               ALL ( SELECT sum( S2.quantity )
                     FROM Sold as S2, Bookstore as B2
                     WHERE S2.name =B2.name AND
                           B1.city = B2.city
                     GROUP BY S2.name );
```

Book(<u>bookid</u>, title, author, publisher, year, price)
Bookstore(<u>name</u>, address, city)
Sold(<u>name</u>, <u>bookid</u>, quantity)

c) Write the equivalent SQL SELECT query for the following relational algebra expression

$$\Pi_{\text{name,numBooks}} (\gamma_{\text{name,sum(quantity)->numBooks}} (\sigma_{\text{publisher='McGrawHill' AND year='2015'}} (\text{Bookstore} \bowtie \text{Sold} \bowtie \text{Book}))$$

Solution:

```
SELECT Bookstore.name, sum(quantity) AS numBooks
FROM Bookstore, Sold, Book
WHERE Bookstore.name = Sold.name AND
      Sold.bookid = Book.bookid AND
      Publisher = 'McGrawHill' AND
      year = '2015'
GROUP BY Bookstore.name;
```

Question 4 (16pts) Relational Design Theory

Consider a relation R(A,B,C,D,E), with FDs
AB → C,
C → A,
C → BD,
D → E

(6pts) (a) List all the keys of R. Do not list superkeys which are not (minimal) keys.

Solution: Keys are : AB ; C

Since E doesn't appear on the left hand side of any FD, it can't drive additional attributes. Any key which includes E, will be a super key. Therefore it is sufficient to look at the attributes subsets that doesn't include E.
{A}+ = {A}
{B}+ = {B}
{C}+ = {ABCDE} ---- key
{D}+ = {DE}
{AB}+ = {ABCDE} ----key
{AC}+ superkey
{AD}+ = {ADE}
{BC}+ superkey
{BD}+ = {BDE}
{CD}+ superkey
{ABC}+ superkey
{ABD}+ superkey
{ACD}+ superkey
{BCD}+ superkey
{ABCD}+ superkey

(10pts) (b) Is this relation in BCNF? If you answer is yes, explain why it is. If you answer is no, decompose the relation into BCNF, showing your decomposition steps.

No. The last FD, D → E, violates BCNF. We decompose into R1(DE) and R2(ABCD).

R1 is in BCNF because the only relevant FD, D → E, does not violate BCNF (and because its a two-attribute relation).
R2 has the relevant FDs AB → C, AB → D, C → A,C → B,C → D, none of which violate BCNF. Therefore we are done.

Name

Question 5 (20pts) Indexing

Consider the following relational schema for a portion of a company database:
Project (pno, proj_name, proj_base_dept, proj_mgr, topic, budget)
Manager (mid, mgr_name, mgr_dept, salary, age, sex)

Manager table stores the information about project managers.
Note that:

- pno is the primary key for Project and mid is the primary key for Manager.
- Project.proj_mgr is a foreign key referencing Manager.mid
- each project is based in some department (proj_base_dept),
- each manager is employed in some department (mgr_dept), and
- the manager of a project need not be employed in the same department (in which the project is based).

Suppose you know that the following queries are the five most common queries in the workload for this university and all five are roughly equivalent in frequency and importance:

1. List the names, departments, and ages of the managers who earn more than $90K. (i.e., salary>$90K)
2. List the names, ages, and salaries of managers of a user-specified sex (male or female) working in a given department. You can assume that, while there are many departments, each department contains very few project managers.
3. List the names, topics and budgets of all projects with managers whose ages are in a user-specified range (e.g., younger than 30).
4. List the departments such that a manager in this department manages a project based in this department.
5. List the project names and their departments with budget more than $900K.
6. List the name of the project with the lowest budget.
7. List the mid's of all managers in the same department as a given project *pno*.

These queries occur much more frequently than updates, so you should build whatever indexes you need to speed up these queries. However, you should not build any unnecessary indexes (or include any unnecessary attributes in an index), as updates will occur (and would be slowed down by unnecessary indexes). Assume index only query plans are possible.

Given this information, suggest indexes that will give good performance for each of the queries above . In particular decide,

(i) which attributes should be indexed (both single- and multiple-attribute index search keys are permitted),
(ii) whether each index should be a clustered index or an unclustered index, and
(iii) whether it should be a B+ tree or a hashed index.

Solution:

Please note that the below solutions overall propose a single clustered index for the Project relation and a single clustered index for the Manager relation. All other indexes are unclustered.

The answer to each question is given below.

Query 1.

- **a clustered B+tree index on *<salary>* for the Manager relation**

We can scan all managers in the clustered B+ tree index and check if their salary is more than $90K. Since the query is a range query, the index need to be clustered and need to be a B++ tree.

Query 2. Either:

- **an unclustered hash index on *<mgr dept,sex>* for the Manager relation.**

Since each department contains very few managers, selection on mgr_dept will return very few tuples. Therefore an unclustered index will work.
(The order of the attributes matter here: mgr_dept is included first since it is more selective)

Query 3.

- a unclustered B+ tree index on *<age, mid>* for the Manager relation, and an clustered hash index on *<proj_mgr>* for the Project relation.

We can do an **index only scan** to find managers whose age is in the specified range, and then hash into the Project relation to get the project names, topics and budgets.

Query 4:

- a unclustered B+ tree index on *< mgr_dept, mid >* for the Manager relation, and an unclustered hash index on *<proj_base_dept, proj_mgr >* for the Project relation.

We can scan all managers in the B+tree and use the hash index on *< proj_base_dept, proj_mgr >* on the Project relation to check if *mgr_dept = proj_base_dept* and *mid= proj_mgr .* (the order of the attributes will not matter in this case)

Solution-2:

Alternatively we can create:

- a unclustered B+ tree index on *<age,mgr_dept,mid>* for the Manager relation, and an unclustered hash index on *<proj_base_dept, proj_mgr>* for the Project relation.

We can use this index for both query 2 and 3.
For query 2: We can do an index only scan to find managers whose age is in the specified range, and then hash into the Project relation to get the project names, topics and budgets.
For query 3: We don't need a new index. We can scan all managers in the B+tree and use the hash index on *<proj_base_dept, proj_mgr>* on the Project relation to check if *mgr_dept = proj_base_dept.*

Query 5:

- unclustered B+ tree index on <*budget,proj_name,proj_base_dept*> for the Project relation.

Since it is a range query, the index need to be a B++ tree. We can make an index only scan in the unclustered B+ tree index to find the first page with budget > $900K and scan through the rest of leaf pages to return the *proj_name* and *proj_base_dept* values.

Query 6:

We can use the unclustered B+ tree index we created for query-5. No additional index is needed. We can go down the tree to find the lowest budget and fetch the name of the project.

Query 7:

- unclustered hash index on <*pno , proj_base_dept*> for the Project relation. We can re-use the index <*mgr_dept, mid*> that is proposed for Query 4. No additional index is needed for Manager.

We can get the proj_base_dept of the project by using the <*pno, proj_base_dept* > index, and then use the B++ tree index on <mgr dept,mid> to get the managers in this department.
(Note that an index on <*pno, proj_ base_dept*> for Project would allow us to do an index only scan on Project. However, since there is exactly one base department for each project (pno is the key) an index on <*pno*> only will work as well. (It does add only one I/O per project.))

Name

Question 6 (12pts)

Consider the relation R(a,b,c,d,e) with the following properties/statistics:

- The primary key is ab (neither a nor b values are unique in R).
- There are a total of **15,000** tuples in R stored in **1,000** pages (when stored in a heap file).
- The number of distinct values of attribute a in R is **300.**
- The "a" values are uniformly distributed in relation R.

Suppose we have a B+tree index available for attribute "a" of R. The height of the B+ tree is 3. Assume each node of the B+ tree index occupies one page. Assume the cost is measured by disk I/Os for accessing the index and the records.

We run the following query on relation R:

```
SELECT *
FROM R
WHERE a = 1000
```

a) (**6pts**) Estimate the average cost of executing the above query when the B+tree index is clustered, i.e., the records with the same a value are stored consecutively in the disk but may spread in different blocks. (Assume the B+tree has 67% occupancy, i.e., the physical data pages are 1.5 times more than original data file.)

# of leaf pages in clustred B+ tree = 1.5 * 1000 = 1500
# of matching records for the query = 15000/300 = 50
# of leaf pages that include matching records = (50 * 1500) /15000 = 5 pages

Cost = height of the clustered B+ tree +
# of additional leaf pages that include matching records

= 3 + (5-1) = 7

b) (**6pts**) Estimate the cost of executing the above query when the index is unclustered. (Assume the size of a data entry in the B+tree is around 20% of the data record (the data entry also includes the RID)).

# of leaf pages in clustred B+ tree = 1500 * 20% = 1500 /5 = 300 pages

pages that include the matching 50 data entries = (300 * 50) /15000 = 1

Cost = height of the clustered B+ tree +
# of additional leaf pages that include matching data entries +
# of data pages read to retrieve matching tuples
= 3 + (1-1) + 50 =53

Name

<u>Question 7</u> (6pts)

Mention four advantages of using database management systems (DBMS) over file systems.

- Data consistency in presence of concurrency
- Reliability in presence of failures and system crashes.
- Efficient associative access to very large amounts of data
- A high level Query language (SQL) to define, create, access, and manipulate data.
- Security and authorization
- Prevention of data redundancy and inconsistencies
- Data abstraction and support for multiple data views