Name: ***Solution S14***

SID: ______________________________

**Midterm II**
**Spring 2014**
**CSC3287 Database Systems Concepts**

# Instructions:

Work alone to complete the questions. This test is an open book/open notes test. Cheating will be dealt with according to department policy.

Put your name on each page to be sure that I know whom to credit for the work.

The test, in total, is worth 100 points.

**I WILL NOT ANSWER QUESTIONS DURING TESTS** because the nature of this work requires you to make assumptions. If you have questions during the test, state your assumption on your paper and follow it through in your answer. Partial credit will be given for stating and following your assumptions even if they are not what I was thinking when the test was written. You may add things to your model if necessary but you must also state your assumptions that go with your addition.

Some questions have multiple answers.

Write your expressions and draw your models in the manner that you are most comfortable with. But be sure to use the formal notation taught in the book or in class and be as complete as possible with your answers.

If you finish early, check for completeness in your drawings and expressions. When you are convinced that you are done, deliver your test in one of two ways:

Email to debra.parcheta@ucdenver.edu
Or
Hand it to me at class on 4/1/2014 at 3:30 PM

Verify that two Exhibits are attached:
Museum Database EER Diagram and
Airline Database System ER Diagram

Name: ***Solution S14***

SID: ___________________________

1. (5 Points) Name an attribute from the Museum Database System that would be appropriate for building a primary index. Tell why you chose that attribute.

Use one of the keys shown in the diagram.

ARTIST.Name OR

ART_OBJECT.A_IDNo OR

COLLECTION.C_Name

2. (5 Points) Name an attribute from the Museum Database System that would be appropriate for building a secondary (non-key) index. Tell why you chose that attribute.

A Title, Name, date or the Epoch of the art would work – any attribute that we might want to group on but is not necessarily a key. A candidate key value would work well, but I don't see any. Duplicate attribute values are allowed.

3. (5 Points) Name an attribute from the Museum Database System that would be appropriate for building a clustering index. Tell why you chose that attribute.

Any foreign key would work – attributes with a dashed line under them in the diagram are:

ART_OBJECT.AtritstName
ARTIST.AA_IDNo
BORROWED_ART.Collection
EXHIBITION.EA_IDNo

4. (15 points) Assume that a B+ Tree has been built using consecutive integers as the values in each node, starting with the integer 1 and ending with the integer 122, where p = 6, n at the leaves is p-1, n for an internal node = p+2, and the tree is 70 to 80% full at the leaves. Calculate the efficient minimum number of block accesses in the resulting B+ tree to retrieve the set of records with values. {6, 18, 62, 100, 101, 106}. Assume that the actual retrieval takes one block access and that, when the record is retrieved, the search for the next record continues from the last visited node. State any other assumptions you make.

**Constructing the tree in the manner indicated in the problem allows me to retrieve the record list in 28 or 29 block accesses.**

Just think it through from what you know about the B+ tree.
Leaves have every value represented in order in a linked list.
Assume 80% full – top of the given range.
I have to store 122 values – how many leaf nodes do I need to store that many values at 80% full where I am told that the leaf holds p-1 (5) data ptrs?

$122/(.8*5) = 30.5$

Well, we don't ever state anything as half a node. (Now it's 31 leaf nodes.)

And I need a balanced tree so an even number of nodes seems proper. (32)

Great – 32 nodes gets it.
Can I fit 32 nodes in the first level (level 0 ) of the tree? No – at most, if it were a leaf node it could hold 5 data values and ptrs to records.
Can I fit 1 in the root and expand to level 1 and fit the rest into 2 more nodes?

Nope – My choices would range from

a) 1 data value without ptrs to records at the root + 5 data values with ptrs in each of the 2 hanging nodes (balanced tree). (This only stores 10 values in 2 leaf nodes – I need 32 leaf nodes.)

To

b) 8 data values (p+2 for internal nodes) without ptrs to records at the root + 5 values with ptrs in each of 9 hanging nodes (balanced!) ( 9 leaf nodes is not enough – I need 32.

I know there will be a third level. If I fill the root, and level 1, the max number of leaf nodes at level 2 is 81 leaves. Plenty to work with.
MY TREE HAS 3 LEVELS, and I am using 32 LEAVES. My root has one value, my level 1 has 8 nodes with 3 data values each, creating 4 level 2 nodes from each giving me the 32 leaves I need in a balanced tree.

Let's access the records we are looking for:

To find 6 – I travel root, level 1, level 2 and to the record = 4 block accesses

To find 18, it's not in the same leaf, so travel back out and then down to the right record through another branch starting (as the problem says, from the last node visited. So I am still at the record node for value 6, I travel back to level 2, back to level 1, back to root and then 3 more accesses to the record with 18, etc….

The whole trip to retrieve all is
Record with 6 —> 4 accesses
Record with 18 → 6 more accesses
Record with 62 → 6 more block accesses
Record with 100 → 6 more block accesses
Record with 101 – maybe 0 more block accesses if it's in the same leaf, but maybe 1 more if I traverse the linked list.
Record with 106 → 6 more block accesses from there. (You may have chosen again to traverse the linked list – I didn't. You would have to state that assumption clearly to get full points for a much lower number.)

IN TOTAL, to retrieve all of the records that was
4 + 6 + 6 + 6 +(0 or 1) + 6 = 28 or 29 blocks accessed.

Answers that were within 1 of the value listed here were considered correct. Other assumptions were considered.
You were not supposed to "break" the parameters given to you.
You had to remember that the n in internal nodes stood for data values that did not point to a record – that's why we can fit more in (no data ptrs).
Telling me the problem could not be done was wrong, but you got credit if you wrote me some reasoning.
Telling me that you can't work with parameters that contradict the average cases that we studied in class is also not acceptable. In the real world, you are unlikely to encounter the average case every time.

Name: ***Solution S14***

SID: ____________________________

5. (10 points) Refer to the Museum Database exhibit.
Use formal relational mapping notation to describe 5 different constraints on the Museum Database. (Any 5 you like.)

You had to describe with formal notation and tuple constraint, key matching constraint, any list constraint….and of course name the attributes and tables shown in the diagram.

From your book, the rules were:

"We will use the following notation in our presentation:▪ A relation schema $R$ of degree $n$ is denoted by $R(A_1, A_2, ..., A_n)$.

The uppercase letters $Q$, $R$, $S$ denote relation names. The lowercase letters $q$, $r$, $s$ denote relation states. The letters $t$, $u$, $v$ denote tuples.

In general, the name of a relation schema such as STUDENT also indicates the current set of tuples in that relation—the *current relation state*—whereas STUDENT(Name, Ssn, ...) refers *only* to the relation schema.

An attribute $A$ can be qualified with the relation name $R$ to which it belongs by using the dot notation $R.A$—for example, STUDENT.Name or STUDENT.Age. This is because the same name may be used for two attributes in different relations. However, all attribute names *in a particular relation* must be distinct.

An $n$-tuple $t$ in a relation $r(R)$ is denoted by $t = <v_1, v_2, ..., v_n>$, where $v_i$ is the value corresponding to attribute $A_i$. The following notation refers to **component values** of tuples:

Both $t[A_i]$ and $t.A_i$ (and sometimes $t[i]$) refer to the value $v_i$ in $t$ for attribute $A_i$.

Both $t[A_u, A_w, ..., A_z]$ and $t.(A_u, A_w, ..., A_z)$, where $A_u, A_w, ..., A_z$ is a list of attributes from $R$, refer to the subtuple of values $<v_u, v_w, ..., v_z>$ from $t$ corresponding to the attributes specified in the list."

Full credit was given for writing at least 2 kinds of these in your 5 statements.

Domains Attributes Tuples and Relations could have been included.

6. (5 points) Referring to the Museum Database exhibit, write the relational algebra expression that produces a list of the COLLECTIONs where the COLLECTION.Type was recorded as "Painting."

LIST_OF_PAINTING_COLLECTIONS $\leftarrow \Pi_{\text{C\_Name}} (\sigma_{\text{COLLECTION.Type = "Painting"}} (\text{COLLECTION}))$

Name: ***Solution S14***
SID: ___________________________

7. (5 points) Refer to the Airline Database system. Name the entity(ies), if any, that would have a fixed record size. Tell why you would know that.

You could site any entity where you assumed that the size of the record contained no variable-sized data types (No varchars).
AIRPLANE, FLIGHT_LEG, LEG_INSTANCE and SEAT are prime candidates for being fixed in size.

8. (5 points) Refer to the Museum Database exhibit. Specify the size of a record in the PERMANENT COLLECTION.

Your answer should have included 11 attributes – some inherited from the parent table – the data type assigned to them and an assumed size. Since the question asked for the record size, you needed to total it.

9. (5 points) Continue to refer to the Museum database. Given a block size of 1024, what is bfr for the records in ART_OBJECT.

If you did the question 8 problem correctly you already know the size of the ART_OBJECT record – it's part of the PERMANENT COLLECTION total.
Use the floor of B/R to determing the bfr.

10. (10 points)
a. Referring to the Airline Database Exhibit, tell me, for each entity what type of record blocking technique would be used and why.

Spanned and unspanned are the two types of record blocking you could choose from. Your reason for using spanning should have something to do with the possibility of R > B being true for the entity. Otherwise, unspanned is appropriate. Many students did not consider record size in relation to block size at all – that was disappointing.

11. (5 points) Given the relational nature of the Airline database, what sort of file allocation would work best and why?

This reminds me of the voice mail system where we group things by their plane number or by a date, or by an airport or an airline. and all of the things associated with one of those aspects may need to be read quickly together. Clustered allocation would work great for this database.

Other answers – linked and indexed – were fine too if you gave the right reasons for thinking they were good.
Contiguous is not appropriate for a database with this many relationships in it.

12. (5 points) Write the generic form of an aggregation statement using Relational Algebra notation.

$$Q_{\text{GroupName, aggregateValueName}} \leftarrow {}_{\text{grouping attribute}}\, \mathfrak{I}_{\text{aggfunction(attribute)}}\,(\text{TABLE})$$

13. (5 points) Write an actual aggregation statement using data from the Airline Database System. (Any aggregation that you can produce is fine.)

See your test for notes – you had to plug attributes from the table you named in the equation into the spots in the equation shown above, making grouping and new value output columns for the results of the aggregation of an attribute.

Name: ***Solution S14***

SID: ___________________________

14. (15 points) Consider the two tables T1 and T2 shown in Figure 6.15 below. Show the results of the following operations on the tables. (Extra blank page provided for answers if needed.)

a. $T1 \bowtie_{T1.P = T2.A} T2$

b. $T1 \bowtie_{T1.Q = T2.B} T2$

c. $T1 ⟕_{T1.P = T2.A} T2$

d. $T1 ⟖_{T1.Q = T2.B} T2$

e. $T1 \cup T2$

f. $T1 \bowtie_{(T1.P = T2.A \text{ AND } T1.R = T2.C)} T2$

---

**Figure 6.15**
A database state for the relations *T1* and *T2*.

**TABLE T1**

| P | Q | R |
|---|---|---|
| 10 | a | 5 |
| 15 | b | 8 |
| 25 | a | 6 |

**TABLE T2**

| A | B | C |
|---|---|---|
| 10 | b | 6 |
| 25 | c | 3 |
| 10 | b | 5 |

a. Table Q

| Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|
| 10 | a | 5 | 10 | b | 6 |
| 10 | a | 5 | 10 | b | 5 |
| 25 | a | 6 | 10 | b | 5 |

b. Table Q

| Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|
| 15 | b | 8 | 10 | b | 6 |
| 15 | b | 8 | 10 | b | 5 |

c. Table Q

| Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|
| 10 | a | 5 | 10 | b | 6 |
| 10 | a | 5 | 10 | b | 5 |
| 15 | b | 8 | NULL | NULL | NULL |
| 25 | a | 6 | 25 | c | 3 |

d. Table Q

| Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|
| 15 | b | 8 | 10 | b | 6 |
| NULL | NULL | NULL | 25 | c | 3 |
| 15 | b | 8 | 10 | b | 5 |

e. Table Q

| Q1 | Q2 | Q3 |
|---|---|---|
| 10 | a | 5 |
| 15 | b | 8 |
| 25 | a | 6 |
| 10 | b | 6 |
| 25 | c | 3 |
| 10 | b | 5 |

f. Table Q

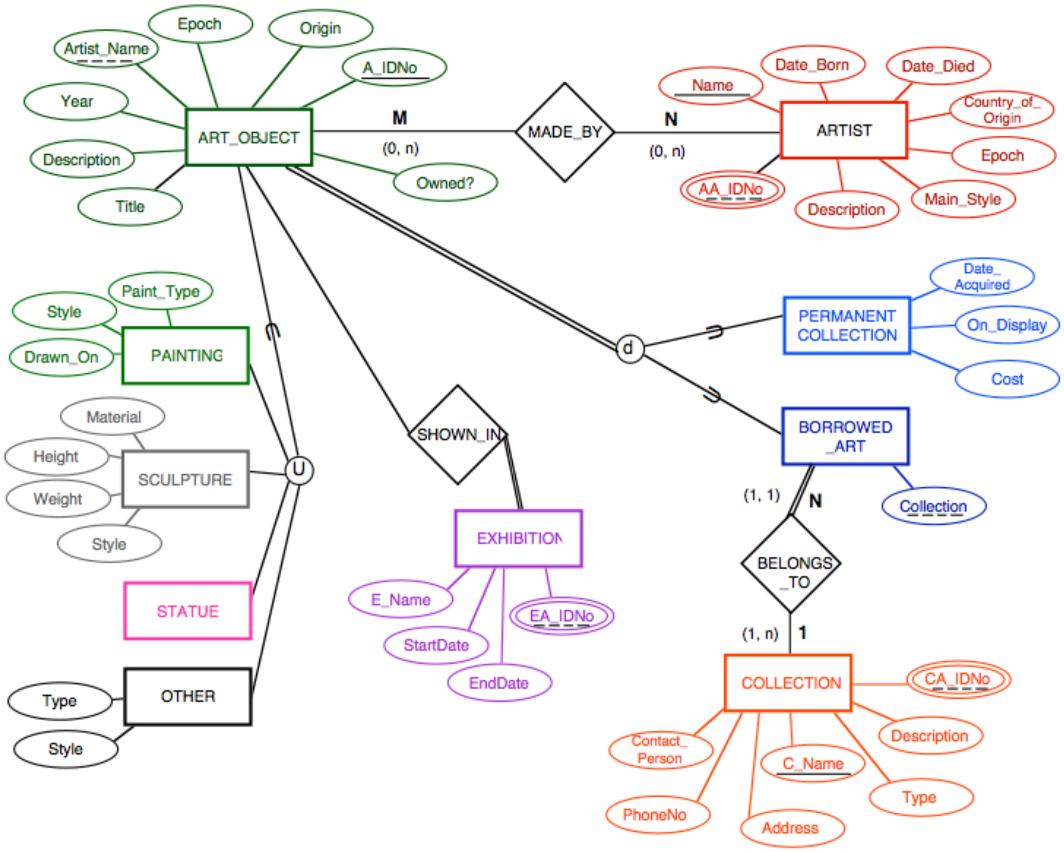
| Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|---|---|---|---|---|---|
| 10 | a | 5 | 10 | b | 5 |

Name: ***Solution S14***

SID: ______________________________

**MUSEUM DATABASE SYSTE**

( One possible solution.)




**Assumptions**:
ARTISTS, COLLECTIONS, EXHIBITIONS have unique names.
ART_OBJECTS have a unique ID number.
ART_OBJECTS are either part of the museum's PERMANENT COLLECTION or are BORROWED_ART
in which case we must collect information about the COLLECTION it came from.
1 or more ARTISTS make 1 or more ART_OBJECTS.
It is possible to record an artist without yet having an ART_OBJECT made by him/her in the database.
It is possible to record an ART_OBJECT without yet knowing who the ARTIST is.
Every BORROWED_ART is borrowed from a COLLECTION that exists in the database.
An EXHIBITION shows 1 or more ART_OBJECTS but an ART_OBJECT does not necessarily have to be on EXHIBITIC
A COLLECTION does not have to loan out its ART_OBJECTS.
A COLLECTION contains 1 or more ART_OBJECTS.

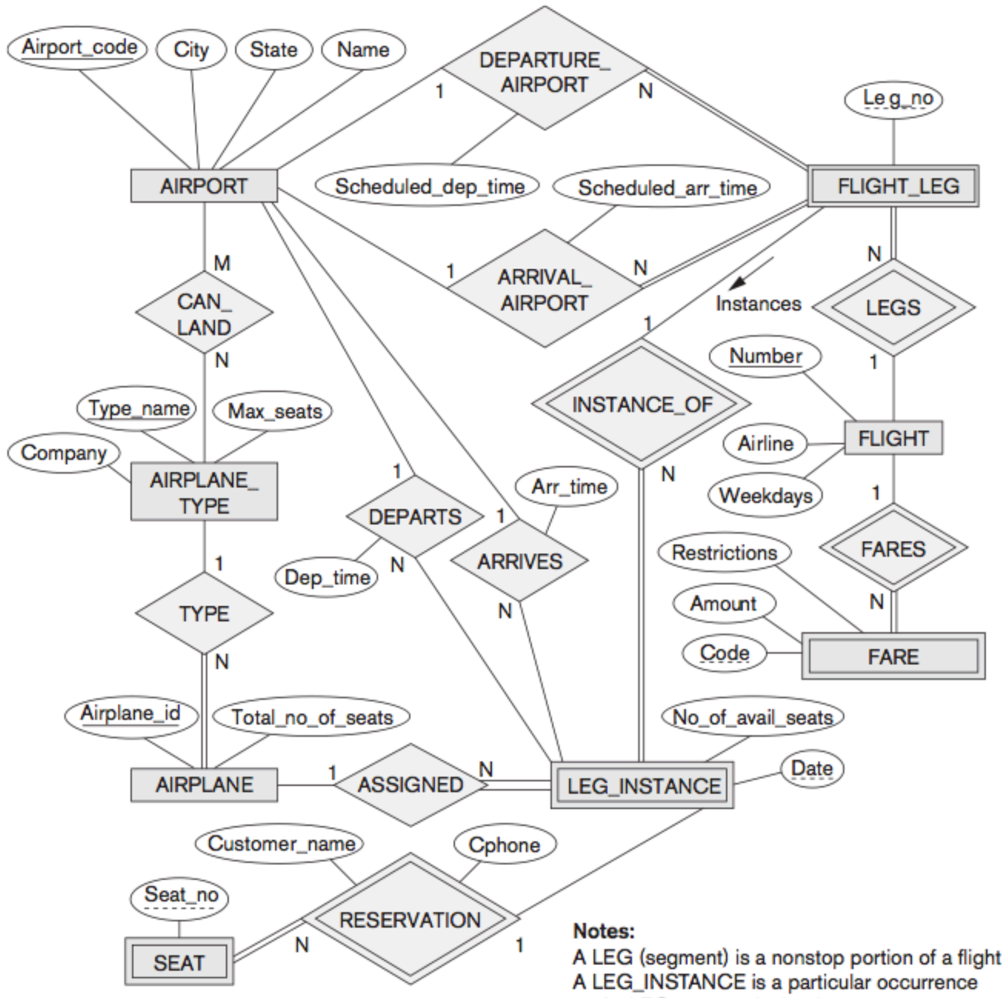
Name: ***Solution S14***

SID: ______________________________



# Airline Database System

**Figure 7.20**
An ER diagram for an AIRLINE database schema.




**Notes:**
A LEG (segment) is a nonstop portion of a flight
A LEG_INSTANCE is a particular occurrence of a LEG on a particular date.