Name____

CS186 Login ____

UNIVERSITY OF CALIFORNIA
College of Engineering
Department of EECS, Computer Science Division

CS186 J. Hellerstein
Spring 2003 Midterm

Midterm Exam: Introduction to Database Systems

This exam has five problems, worth different amounts of points each. Each problem is made up of multiple questions. You should read through the exam quickly and plan your time-management accordingly. Before beginning to answer a question, be sure to read it carefully and to *answer all parts of every question!*

Except where specifically directed otherwise, you **must** write your answers on the **answer sheets** provided. You may use the sheets with questions as scratch paper. You also must **write your name** at the top of **every page**, and you must turn in all the pages of the exam. **Do not** remove pages from the stapled exam! Two pages of extra answer space have been provided at the back in case you run out of space while answering. If you run out of space, be sure to make a "forward reference" to the page number where your answer continues.

Good luck!

1. Disk and Buffers

The specifications sheet for the Seagate Cheetah X15-36LP 40 GB disk says that it has a sector size of 512 bytes, about 80 million ($8x10^{10}$) sectors, about 20,000 ($2x10^{5}$) cylinders, 4 double-sided platters, and an average seek time of about 4 msec. The disk platters rotate at about 18,000 ($18x10^{5}$) rpm (revolutions per minute), and one track of data can be transferred per revolution. We will put our database on this disk. The buffer pool of this DBMS consists of buffer pages with 1024 bytes each. The disk block size is the same as buffer page size.

If your arithmetic is getting messy in this question, you can leave your answer unsimplified, as an equation of 0 variables (e.g. "9467/32 + 212").

a) Based on the information above, how many tracks are there on each platter of this disk?

b) Suppose that a table containing 10,000 records of 100 bytes each is to be stored on such a disk, and no record is allowed to span two blocks. How many blocks are required to store the entire table? (Hint: first compute the number of records per block! You may assume that the disk blocks are arranged for fixed-length records)

c) Suppose the entire table in (b) is to be read from this disk. List each of the three components of the access time for reading the table, and give average values for each of these components. (Hint: you may need to compute the maximum *transfer rate* of the disk in bytes per second. To do so, assume that each track holds the same amount of data, and the time to move to the "next" track is negligible.)

d) Suppose that another table of the same size as the table in (b) is to be nested-loop joined with the table in (b). How many buffer pages (at least) do we need to avoid sequential flooding? Assume that the buffer replacement policy is LRU and all the buffers are unpinned and put in the free list before executing the join. Also assume we are writing the output to disk.

Name____

Answers to Problem 1

2. Hashing

In this question, you are implementing a database to handle applicants to be contestants on the TV show "The Bachelorette". Consider the relation *Applicants(name, age, IQ, haircolor, homestate, walletthickness).*

We are interested in writing queries for aggregation and duplicate elimination.

a) When you set up the entry form, you let people type any string they wanted for the haircolor field. Some weird stuff came in! Values included "fuschia", "dishwater" and "noneofyourbusiness". Seems like a lot of the unusual entries came from Florida and California. Since you're curious, you ask the following query (Q1):

```
    select   homestate, count(distinct haircolor)
      from   Applicants
  group by   homestate
```

Would the hash-based grouping you implemented in Homework 2 work for this query? Why or why not? Would sorting work better?

You are interested in choosing people with a variety of attributes for the show. **Consider the following query (Q2) for the rest of the question:**

```
select  distinct name, haircolor, homestate
  from  Applicants
```

b) Let us assume a hash-based approach to duplicate elimination. Further:

- Let Applicants be N blocks big.
- Let the memory allocated for hashing be H buffers
- Let u be the number of unique tuples in R
- Let b be the number of tuples that fit on a single disk block

Assume that you have a naïve implementation of hashing, which is unable to spill to disk (i.e. the code in Postgres before you added your HW2!) What is the largest value of N that can be handled efficiently in this implementation? Why?

c) Assume a simple, 2-phase disk-based hashing strategy as described in lecture (partition+rehash – not hybrid hashing!). Give an expression for the number of I/Os for processing duplicate elimination. Do not count any I/Os for reading the input to the partitioning phase, or writing the output of the rehashing phase!

d) Recall the hybrid hashing strategy you implemented in Homework 2. We made the approximation that the output buffers for each partition were not accounted for in the H buffers available for hashing. When the number of partitions is very high, this can be a prohibitive memory usage. A more rigorous strategy would be to use a fixed number of partitions. In such a scheme:

- Let the number of on-disk partitions be P
- In phase 1, let the in-memory hash table have H-P buffers
- In each subsequent phase the in-memory hash table has H-1 buffers

Assume that the data in R is uniformly distributed, and we have "perfect" hash functions that ensure that each partition has an identical number of unique tuples. What is the maximum number of unique tuples (u) in R that can be accomodated by this scheme ? Also, give an expression for the maximum number of I/Os to process the query Q2 with this scheme.

e) Suppose that there is no way to predict the maximum number of unique tuples (u) in the table R. In 10 words or less, say how you would modify the scheme in (d) above to still work.

Name____

Answers to Problem 2

Name____

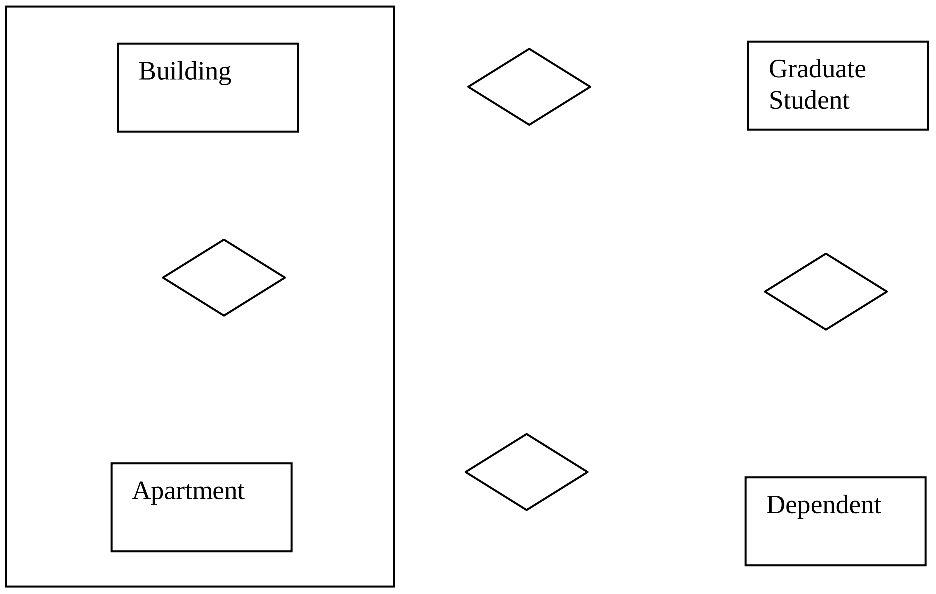
3. ER Diagram (70%, 30% + 30% + 10%)

The UC Berkeley housing office has decided to computerize its information on graduate students and their dependents living in the married housing apartments at Albany Village.

Here are the rules of the database:

- Graduate students are identified by their student ID (SID). They also have name, age, sex and department as attributes.
- Graduate students have dependents (with name and age as attributes). We are not interested in information about a dependent once the graduate student leaves. Assume that only one member of the family is a graduate student.
- Each graduate student is assigned to some apartment within the village. Most of them live in their own apartments, but there are some of them who share the same apartment with others. Some apartments can be empty.
- Each apartment exists within a building and can be identified by an apartment number unique to the building. All apartments are assigned to exactly one building, and apartment numbers may be reused across buildings.
- All buildings in the village are assigned a unique number. Each building must have at least one apartment.
- Each building has a building manager who is in charge of maintenance issues. Each building must have exactly one building manger. The building manager is one of the graduate students who live within the building.

a) Complete the following Entity-Relation diagram based on the information above – *draw directly on the diagram below*. Make sure to make the distinctions among your lines clear! Don't forget to underline your keys as appropriate, and label your relationships.



b) Using the blank spaces provided just below, write down a relational schema for the rules above, and identify all foreign keys. As an example of our notation, we provide a dummy answer for table Foo with two attributes x and y, where y is a foreign key to column Z of table Baz, and y cannot be null. (*Notes:* If your answer to (a) is correct, then this schema will be consistent with (a). But you needn't answer (a) to answer this! Also, you do not have to fill in all rows of the table below!! We provided more rows than needed.)

Table name	**Attributes**	**Primary Key**	**Foreign Keys**
Foo	x,y	x	Y reference Baz (Z), not null

c) Write a single SQL query for this system to find the graduate students that either have no dependents, or have dependents older than 5. Output distinct SIDs, names and departments of graduate students.

d) There may be some rules that we have stated above for the database that cannot be captured by your relational schema, and would require the use of table constraints or assertions. Identify in words **any one** of these rules. If you think there are none, indicate so.

Name____

Answers to Problem 3 (parts c - d)

Name____

4. **Relational Algebra and Calculus (30%, 15% + 15%)**

We revisit the NBA schema from hw3.

Player (playerID: integer, name : varchar(50), position : varchar(10), height : integer, weight : integer, team: varchar(30))
Each Player is assigned a unique *playerID*. The position of a player can either be *Guard*, *Center* or *Forward*. The height of a player is in inches while the weight is in pounds. Each player plays for only one team. The *team* field is a foreign key to Team.

Team (name: varchar(30), city : varchar(20))
Each Team has a unique name associated with it. There can be multiple teams from the same city.

Game (gameID: integer, homeTeam: varchar(30), awayTeam : varchar(30), homeScore : integer, awayScore : integer)
Each Game has a unique *gameID*. The fields *homeTeam* and *awayTeam* are foreign keys to Team. Two teams may play each other multiple times each season. There is a check constraint to ensure that homeTeam and awayTeam are different.

GameStats (playerID : integer, gameID: integer, points : integer, assists : integer, rebounds : integer)
GameStats records the performance statistics of a player within a game. A player may not play in every game, in which case it will not have its statistics recorded for that game. *gameID* is a foreign key to Games. *playerID* is a foreign key to Player. Assume that two assertions are in place. The first is to ensure that the player involved belongs to either the involving home or away teams, and the second is to ensure that the total score obtained by a team recorded (in Game) is consistent with the total sum (in GameStats) of individual players in the team playing in the game

For each of the following parts, you can get partial credit (50%) if you choose to write SQL instead of relational algebra or calculus. Feel free to use compound relational algebra operators, such as division.

Write the **relational calculus** for the following. Feel free to output extra columns if that makes your query simpler.

a. Find the tallest player(s) from each team.
b. List the players who played in all away games for their team.

Write the **relational algebra** for the following. Be sure to get exactly the output columns requested.
a. List each player's playerID, and the gameIDs where they earned "triple doubles". (A "triple double" is a game in which the player's number of assists, rebounds, and points are all at least in the double-digit range or higher).
b. List all names of Chicago Bulls players who have played in all Bulls games.

Name____

Answers to Problem 4

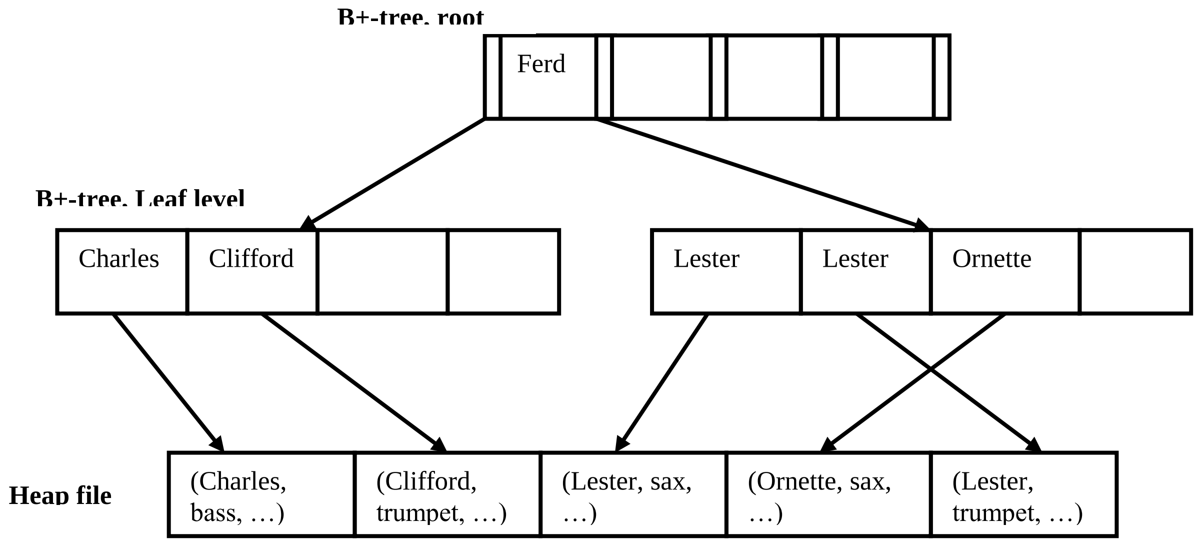
5. **Indexing**

Consider the following picture of a *clustered* B+-tree index and heap file on the relation *Artists*. The first column of the table in the heap file is called *FirstName*, and is indexed by the tree; the second column *Instrument* is also shown, but the remaining columns have been omitted from the picure (they're the "…" in each tuple.) Assume that the *FirstName* column is a fixed-length field – all entries are padded to 10 characters.

Each block in the index is shown as a large rectangle – there are currently three of these. Each block shows both the entries in the block, and the free space where more entries can fit.

In the heap file, each tuple is an entire block long. Hence each rectangle at the heap file level of the picture is a disk block of its own.

a) What is the *order* of this index?
b) Suppose the index were an ISAM instead of a B-tree. Give a query (in SQL, or in relational algebra or calculus) that would be more efficient in the ISAM index than in a B+-tree.
c) Give a sequence of operations on the database that would cause the tree to look the way it does. Be sure to include the initial construction of the index as one of your operations! If you like, you can use SQL commands, but quick English descriptions of each operation will be fine.



Name____

Answers to Problem 5

Name____

Additional Space

Name____

Additional Space