Computer Science 425
Fall 2004
First Take-home Exam
Out: 2:50PM Monday Nov. 8, 2004
Due: 5:00PM SHARP Thurs. Nov. 11, 2004

Instructions: This exam must be entirely your own work. Do not consult with anyone else regarding this exam. If you have questions about what is being asked, contact the graduate teaching assistant, Sumeet Sobti. (Prof. LaPaugh will be out of town and away from email during the duration of this exam.) You are allowed to use the following reference materials: You may use your personal notes and problem set submissions, *Database Management Systems* by Ramakrishnan and Gehrke, the solutions to odd-numbered exercises provided by the authors at `http://www.cs.wisc.edu/~dbbook`, and copies of any of the material on the Web site for the course (staying within the site http://www.cs.princeton.edu/courses/archive/fall04/cos425/...). *No other materials are allowed.* Also you are *NOT allowed to use online database interfaces.*

There are 7 problems, with point values indicated, totaling 100 points. *Be sure to give explanations for your answers.*

On the cover page of your exam submission, write and sign the acknowledgement of original work:

"This exam represents my own work in accordance with University regulations."

Turn in your exam by 5:00pm Thurs. Nov 11, 2004 to the course secretary, Mitra Kelly, in room 323 of the Computer Science Building. Sumeet Sobti will be available during the week to answer questions. His email address is sobti@cs.princeton.edu and his office is room 103C of the Computer Science Building.

The following database will be used in Problems 1, 2 and 3. This database holds information about movies and their distribution to movie theaters.

- The database keeps track of all movies currently available to be shown in theaters. Each movie is identified by its name, producer, and release date. Other information recorded in the database about each movie is its rating, in which movie theaters, if any, it is currently being shown, the actors appearing in the movie, and which local distributors are distributing the movie.
- The database keeps information on each movie theater currently operational. Each theater is identified by its name and location. The manager of the theater and the number of screens (at least one) in the theater are also recorded. For each movie showing in the theater, both its identifying information and the local distributor providing the movie are recorded. Each theater may use several distributors, but only one distributor provides any one movie. A movie house always has as many movies showing as it has screens. The start date and end date of any movie's showing in the theater are also recorded.
- The database keeps information on movie reviews. For each review the following information is kept: the movie reviewed, the reviewer, the date the review was written, the publisher of the review (e.g. NY Times, PBS, MSNBC), and the text of the review. (Even if a review appears in several forms, e.g. Web, newspaper, TV, it has only one publisher.)
- The business name, proprietor, address and telephone number of each local distributor are recorded. Local distributors are uniquely identified by their business names. The movies each distributor carries for distribution are recorded.
- Information is kept on actors, specifically each actor's name, mandatory *Actors' Equity* unique ID number, agent, and a scalar measure of the actor's ability to draw viewers. An actor need not have an agent, but cannot have more than one agent.
- Information is kept on agents who can represent actors, specifically each agent's name, mandatory unique taxpayer ID, address, telephone number and email address.

Problem 1 (5 points)
Identify all the constraints on data or on the relationships between data in the database.

Problem 2 (15 points)
Draw an ER diagram describing the database. Represent the constraints on data and the constraints on relationships between data. Are there any constraints you cannot represent? If so, why? Your ER diagram will be judged not only on whether it correctly captures the database through entities and relationships, but whether it does so in a way that minimizes redundancy in the representation.

Problem 3(15 points)
Translate your ER diagram from Problem 2 into a relational schema. Describe the relational schema using SQL statements to create the relations. Also indicate any primary key, candidate key and foreign key constraints. If there are any constraints that require SQL CHECKs or ASSERTIONs, express them in SQL. (Note: minor SQL syntax errors will not be penalized. Therefore, the prohibition on using an online database interface should not be a hardship.)

Problems 4 and 5 refer to the database holding information for a service providing seeing-eye dogs that was defined in Problem 2 of Problem Set 1. SQL definitions of the relations can be found in the solutions to Problem Set 1. The relational database definitions are repeated here in the generic relational specification style. Primary key fields are underlined:

- relation *Client* with attributes (<u>*Name*</u>, <u>*Address*</u>, *Gender, BirthDate, Impairment*)
- relation *Trainer* with attributes (<u>*SSN*</u>, *Name, Address, NumberYearsService, License*).
- relation *Dog* with attributes (<u>*ID*</u>, *Breed, BirthDate*)
- relation *Trained_Dog* with attributes (<u>*ID*</u>, *CertificationTrainerSSN, CertificationDate*) where foreign key *ID* references *Dog(ID)* and foreign key *CertificationTrainerSSN* references *Trainer(SSN)*
- relation *Trainer_Client_Relation* with attributes (<u>*TrainerSSN*</u>, <u>*ClientName*</u>, <u>*ClientAddress*</u>) where foreign key *TrainerSSN* references *Trainer(SSN)* and foreign key *(ClientName, ClientAddress)* references *Client(Name, Address)*
- relation *Client_Dog_Relation* with attributes (<u>*ClientName*</u>, <u>*ClientAddress*</u>, *DogID*) where foreign key *(ClientName, ClientAddress)* references *Client(Name, Address)*, foreign key *DogID* references *Trained_Dog(ID)*, and *DogID* is a candidate key.
- relation *Trainer_Dog_Relation* with attributes (<u>*TrainerSSN*</u>, <u>*DogID*</u>) where foreign key *TrainerSSN* references *Trainer(SSN)* and foreign key *DogID* references *Dog(ID)*

Problem 4 (20 points)
Express the following queries with relational algebra expressions. You may use any relational algebra operations, including joins and division.

A. Find the IDs and birth dates of all dogs that are currently working with a client whose impairment is 'total'. (Note that Client_Dog_Relation records only current placements of dogs with clients.)

B. Find the SSNs of all trainers who have certified a dog or have more than 2 years with the service.

C. Find the SSNs, names and addresses of all trainers who have worked with all breeds of dogs currently represented by dogs in the database.

Your solutions will be graded primarily on correctness. However, you may lose some points for a solution if it is inefficient – that is, if the solution uses more operations than necessary – especially if it is fairly obvious that some operations are unnecessary. DO NOT spend time trying to optimize your expressions, but do ask yourself if all the operations serve a purpose.

Problem 5 (13 points)

A. Express Problem 4, **query C** in the **domain** relational calculus.

B. Write a SQL query to list the number of trained dogs of each breed. (If there is no trained dog of a particular breed, the breed need not be listed.) (Minor SQL syntax errors will not be penalized.)

Again your solutions will be graded primarily on correctness, but you may lose some points for inefficiency.

Problem 6 (20 points)
In this problem we consider B+ trees on a file that has large records: one record requires a page to store. In contrast, the order of the B+ tree is d=16 and each interior node of the B+ tree is stored in one page; therefore up to 32 search keys and 33 record pointers fit in one page.

A. First consider a B+ tree with data entries that are the actual data records, i.e. each leaf of the B+ tree holds exactly one data record (Alternative 1 in the terminology of *Database Management Systems* by Ramakrishnan and Gehrke). Note that because each data record is so large, leaves can only be full, not between half full and full. In all other ways, this B+ tree follows all the B+ tree rules. Document in detail the precise number of disk page reads and writes necessary for the insertion of a record **in the best case**. Assume the height of the B+ tree is h_1.

B. Now consider a B+ tree with data entries that are pointers to the actual records, i.e. each leaf of the B+ tree holds a set of (*search key value, record id*) pairs (Alternative 2 in the terminology of *Database Management Systems* by Ramakrishnan and Gehrke). Assume 32 such pairs can fit, and leaves are kept at least half full (as we are accustomed to).

i. What can you say about the height, h_2 of this B+ tree in comparison to h_1? Justify any conclusions you make.

ii. What is the best-case cost of an insert with this B+ tree? As in Part A, document in detail the precise number of disk page reads and writes necessary for the insertion of a record **in the best case**.

C. Discuss the pros and cons of each of the alternatives (represented by parts A and B) for building a B+ tree for this file.

Problem 7 (12 points)
Suppose a linear hashed file is in the state shown below just after inserting a record r with hash value $h(r)$=320. The numbers listed for each bucket are the hash values of the hashed records in that bucket, i.e. $h(r)$ for record r. Each page can hold up to 6 data entries.

```
level = 0 (starting with 2 bits of hash)

                  bucket 000:  1000,  88,  616, 200, 576, 320
NEXT TO SPLIT -> bucket  01:   393, 413, 1005, 201, 353, 765   ->
                                          overflow page: 669, 801
                  bucket  10:   402, 222
END (level 0) -> bucket  11:   791, 447, 155
                  bucket 100:  1004, 332
```

A. Assuming the splitting of a bucket is triggered whenever a new overflow page is added and given the configuration shown, what can you say about the number of inserts since the first bucket split (i.e. since bucket 00 split into bucket 000 and bucket 100)?

B. Show the new state of the linear hashed file after a data entry with hash value 264 is inserted.