**Computer Science 425**
**Fall 2006**
**First Take-home Exam**
**Out: 2:50PM Wednesday Oct. 25, 2006**
**Due: 5:00PM SHARP Friday Oct. 27, 2006**

**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 Prof. LaPaugh. 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/openAccess/thirdEdition/supporting_material.htm`

and copies of any of the material on the Fall06 COS 425 Web site for the course (staying within the site

`http://www.cs.princeton.edu/courses/archive/fall06/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 Friday October 27, 2006 to Professor LaPaugh at her office in Room 304 of the Computer Science building. If you wish to turn in your exam when Professor LaPaugh is not in her office, you may give it to the course secretary, Mitra Kelly, in Room 323 of the Computer Science Building or you may put it in Professor LaPaugh's mailbox on the second floor of the Computer Science Building. If you put it in her mailbox, please also email her that it is there.

The following database will be used in Problems 1, 2 and 3. This database holds information about employees of a small digital electronics company called Digital425.

- The database keeps track of all Digital425 employees. Each employee is identified by social security number. The name and address of each employee is also recorded.
- While there are support personnel, the principal employees of Digital425 are designers: hardware designers, software designers, and graphic designers. Additional information to document competence is required for designers: Hardware designers must have a license number. Software designers must have a Bachelors or Masters degree in computer science from an accredited school; the school and degree are recorded. Graphic designers must have had their portfolios certified by the head of the graphics department; the date of certification is recorded. Despite the fact that designers may be multi-talented, Digital425 will only allow each designer to be designated as one specific type: hardware, software or graphic.
- Each product is developed with a team led by one hardware designer, one software designer and one graphic designer. Because team leaders must be carefully matched, the company has put together in advance "leadership groups", which are triples of team leaders (one hardware designer, one software designer and one graphic designer). Each leadership group is recorded with an identifying number. To avoid over-work, any one designer can be assigned to no more than 3 leadership groups.
- Digital425 also uses outside service providers (companies providing services to Digital425). These are also recorded in the database. Each provider is identified by the company name; the address is also recorded. Each provider must provide one or more liaisons to Digital425. For each liaison from a provider company, Digital425 must have the liaison's name and the liaison's telephone number at the provider company.
- Specific Digital425 employees are assigned to interact with specific service provider liaisons. These assignments are recorded in the database.
- Some graphics designers are very particular about the service providers who do their packaging. Therefore, each graphic designer may specify one specific service provider to do all his or her packaging. This request is recorded in the database with the date of the request.

**Problem 1** (7 points)
Identify all the constraints on data or on the relationships between data in the database.

**Problem 2** (20 points)
Draw an entity-relationship (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 primarily on whether it correctly captures the database through entities and relationships. In addition, the grading will consider whether it captures the database in a way that minimizes redundancy in the representation.

**Problem 3**(20 points)
**Part a** (16 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.

**Part b** (4 points) Are there constraints that you cannot specify using only domain constraints and primary key, candidate key and foreign key constraints? If so, for each such constraint, if possible, express it using SQL CHECKs or ASSERTIONs; *if not possible, say why.* (Be sure to say in which table a CHECK appears.)

*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, 5, 6 and 7 will refer to the following relational database that records information about foot races in the United States. We will refer to the database as the "foot races" database. For simplicity, we will assume all runners live in the United States. Primary key fields are underlined; foreign keys are indicated with keyword "references".

- relation *runners* with attributes (*runner ID, name, street address, city, state, emergency contact*)
- relation *courses* with attributes (*course name, length, city, state* ).
- relation *races* with attributes (*race name, date, course name*), where (*course name*) references *courses*.
- relation *results* with attributes
  (*runner ID, race name, date, time*) where (*runner ID*) references *runners* and (*race name, date*) references *races*.
  Attribute *time* records the time it took the runner to complete the race.

The type of every attribute except *length* and *time* is CHAR(100); *length* and *time* are of type REAL.

**Problem 4** (24 points)
Express the following queries with relational algebra expressions. You may use any relational algebra operations, including joins and division. 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.

A. Find the name, length, and city of all courses in Hawaii that are at least 4 kilometers long. (Assume all course lengths are given in kilometers.)

B. Find the runner IDs of all runners who have run at least two races.

C. Find the IDs, cities, and states of all runners who have run a race in the city where they live. (Warning: there may be only one New York City, but there are a lot of Middletown's out there.)

D. Find the names of runners who have run all courses.

**Problem 5** (12 points)
Express the following queries in the **tuple** relational calculus. Again your solutions will be graded primarily on correctness, but you may lose some points for inefficiency.

A. Find the runner IDs of all runners who have run at least two races.

B. Find the names of runners who have run all courses.

**Problem 6** (8 points)
Assume the "foot races" database has been created in a SQL database system with tables exactly matching the specification above. Write a SQL query that returns for each runner, identified by *runner id*, his or her average time on 5-kilometer races. Runners who have not run any 5-kilometer races need not appear in the result. Again your solutions will be graded primarily on correctness, but you may lose some points for inefficiency.

**Problem 7** (9 points)
Show how to represent **just the** *races* **and** *courses* **relations** of the "foot races" database in an XML document. **Do not give a formal schema.** Instead, present your representation by producing the XML document for the following small instance of the "foot races" relational database:

courses:

| course name | length | city | state |
|---|---|---|---|
| olympic_1 | 5 km | Los Angeles | Ca |
| PU_course | 3 km | Princeton | NJ |

races:

| race name | date | course name |
|---|---|---|
| All_state_HS | June 20, 2006 | PU_course |
| Jr_finals | August 7, 2006 | PU_course |
| NCAA_finals | May 5, 2006 | PU_course |
| world_finals | June 1, 2005 | olympic_1 |
| world_finals | June 1, 2006 | olympic_1 |

Does your XML documentation capture all the information (including constraints) in the relational model? If not, what is missing?