University of California, Berkeley
College of Engineering
Computer Science Division - EECS
Prof. Michael J. Franklin

MIDTERM
Fall 2002
CS 186 - Introduction to Database Systems

General Information:

This is a **closed book** examination - but you are allowed one 8.5" x 11" sheet of notes (double sided). You should try to answer as many questions as possible. **Partial credit will be given.** There are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time-consuming than others.

Write all of your answers directly on this paper. **Be sure to clearly indicate your final answer** for each question. Also, be sure to state any assumptions that you are making in your answers. **GOOD LUCK!!**

Question 1 - Functional Dependencies [5 parts, 20 points total]:

a) [3 points] Your cousin, who is studying at a Bay Area university often noted for its sports programs, claims to have learned "Lance's Axioms" for reasoning about Functional Dependencies. He claims that one such axiom, is

Reversitivity: if "A->B" then "B->A"

Give an example relation instance that proves that *Reversitivity* is invalid. (for simplicity, use integers and/or single letters as your attribute values)

b) [4 points] After viewing your counterexample for part (a), your cousin launches into a review of the final scores of the "Big Game" over the past decade. He then claims that there is another axiom:

Kindatransitivity: if "A->C" and "AB->C" then "B->C"

Give an example relation instance that proves that *Kindatransitivity* is invalid.

c) [5 points] It turns out however, that your cousin almost got it right. In fact, there is a rule as follows:

Pseudotransitivity: if "A->B" and "BC->D" then "AC->D"

Using Armstrong's Axioms (not Lance's) show that *Pseudotransitivity* is in fact valid (Be sure to indicate which axiom you are using in each step).

d) [2 points] Why can't you use an example relation instance to answer part "c", as you did for parts "a" and "b" of this question? (be concise - i.e., one or two sentences is sufficient).

e) [6 points] Consider the relation schema **R = ABCDE** and the following functional dependencies on R:

A->D
BC->E
D->AB

R has two candidate keys. What are they (circle your answer)?

Question 2 - Normalization [7 parts, 15 points total]

Consider the relation schema **S = ABCD** and the following functional dependencies on **S**:

A->BCD
B->C
CD->A

For each of the following short questions, be sure to briefly explain your answer.

a) [3 points] S is not in BCNF, but is it in 3NF? Explain your answer.

b) [2 points] Consider the decomposition of **S** into **S1** = ABC and **S2** = BCD. Is this a valid decomposition into BCNF? Explain.

c) [2 points] Is the decomposition of **S** into **S1** and **S2** lossless? Why or why not?

d) [2 points] Is the decomposition of **S** into **S1** and **S2** dependency preserving? Why or why not?

e) [2 points] Consider the decomposition of **S** into **S3** = ABD and **S4** = BC. Is this a valid decomposition into BCNF? Explain.

f) [2 points] Is the decomposition of **S** into **S3** and **S4** lossless? Why or why not?

g) [2 points] Is the decomposition of **S** into **S3** and **S4** dependency preserving? Why or why not?

Question 3 - Formal Relational Languages [5 parts, 20 points total]

Consider the following schema for the Australian Competitive Team Boomerang League. The database tracks information about teams, players, and the outcomes of games they have played. Each game consists of a "home" team and an "away" team. The ACTBL requires that no games can end in a tie. The winner of a game is (obviously) the team with the most points. Assume that every team has played at least one game at home and at least one game away so far. The schema is as follows (primary key for each relation is underlined):

> **TEAMS (TName, TCity)**
> **PLAYERS (PName, Team, Salary)** [where Team is a foreign key referencing TEAMS]
> **GAMES (HomeTm, AwayTm, Date, HomePts, AwayPts)** [where HomeTm and AwayTm are foreign keys referencing TEAMS]

Express the following queries in the indicated query language. Feel free to use the compound relational algebra operators, such as division:

a) [4 points] (Relational Algebra): List the player name and salary of each player who is on a team that has beaten the "Wombats".

b) [4 points] Write the query for part "a" in **Relational Calculus**:

c) [4 points] (Relational Algebra): List the names of all the teams who have won *every* game that they have played at home.

d) [4 points] Write the query for part "c" in **Relational Calculus**.

e) [4 points] (Relational Calculus) List the names of all the teams who have hosted games against *every* team from Sydney.

Question 4 - SQL [6 parts, 25 points total]

Consider the schema from question 3:

> **TEAMS (TName, TCity)**
> **PLAYERS (PName, Team, Salary)** [where Team is a foreign key referencing TEAMS]
> **GAMES (HomeTm, AwayTm, Date, HomePts, AwayPts)** [where HomeTm and AwayTm

are foreign keys referencing TEAMS]

Express the following queries in SQL.

a) [3 points] List the player name, team name, and salary of each player who is on a team that has beaten the "Koalas" while playing at home. The list should not contain duplicates.

b) [3 points] List the names of all the teams who have won *every* game that they have played at home.

c) [4 points] List the names of all the teams who have hosted games against *every* team from Sydney.

d) [4 points] Print a list containing the team name, team city, number of players, and the total payroll (i.e., the sum of all the salaries), for each team that has more than 10 players. Sort this list in decreasing order of total payroll.

e) [8 points] Print the name of the team that has won the most home games. (*Hint:* You might find it easier if you use views).

f) [3 points] Under what conditions would "**TEAMS** LEFT OUTER JOIN **PLAYERS**" have higher cardinality (i.e., more tuples) than "**TEAMS** JOIN **PLAYERS**"? (be concise - one sentence short sentence should suffice).

Question 5 - Data Models [4 parts, 20 total points]

a) [3 points] Draw an ER diagram for an ISA hierarchy in which there are at least two subtypes. Be sure that each of your entities has at least two attributes and indicate any keys.

b) [4 points] For the diagram you drew in the previous part, show two substantially different mappings of this into SQL DDL statements, and briefly describe the tradeoffs between these two.

c) [3 points] Briefly describe how relational "views" could help make query writing easier for the tables that result from one of of your ISA mappings (tell us which one you're using!). Write one (or more if you need it) SQL Create View statement(s) to demonstrate this.

d) [10 points] Draw the ER diagram that represents the schema on the next page. Be sure that all contraints, attributes, and keys in the DDL are shown in your diagram. *NOTE you may find it helpful to remove the last page so you can see the schema while you are drawing. You may also want to practice on a different sheet so that your final answer is readable by the graders.* ***Be sure that your regular and bold lines are clearly distinguishable from each other.***