University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Fall 2002 Prof. Michael J. Franklin

MIDTERM AND SOLUTIONS

CS 186 Introduction to Database Systems

NAME:_____Norm L. Form____________________ **STUDENT ID:**___**00000001**_____

IMPORTANT: `Circle the last two letters of your class account:`

`cs186 a b c d e f g h i j k l m n o p q r s t u v w x y z`

`a b c d e f g h i j k l m n o p q r s t u v w x y z`

DISCUSSION SECTION DAY & TIME:__Sun 7:30am TA NAME: _G something

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!!!

Problem	Possible	Score
1. Functional Dependences	20	20
2. Normalization	15	15
3. Formal Relational Languages	20	20
4. SQL	25	25
5. Data Modeling	20	20
TOTAL	100	100

Name:______________________________ SID:____________________

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)

A	B
1	2
3	2

- need to show a B value with 2 different A values
- If 2 A's have same value, they must have same B value
- not necessary to show two tuples with same A value

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.

A	B	C
1	2	3
2	2	4

- need to show a B value with 2 different C values
- If 2 A's have same value, they must have same C value
- if 2 ABs have same value, they must have same C value
- not necessary to show two tuples with same A value

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).

- A → B given;
- BC → D given
- AC → BC Augmentation of 1 with C
- AC → D Transitivity of 3 and 2

Name:_______________________________ SID:_____________________

Question 1 (continued)

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.)

Because F.D.s must hold across all legal instances of a relation, so showing one example isn't sufficient.

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

$$\mathbf{A \rightarrow D}$$

$$\mathbf{BC \rightarrow E}$$

$$\mathbf{D \rightarrow AB}$$

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

- CD and CA

Name:______________________________ SID:____________________

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

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

$$\mathbf{A \rightarrow BCD}$$

$$\mathbf{B \rightarrow C}$$

$$\mathbf{CD \rightarrow A}$$

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.

Yes, because CD and A are candidate keys. B → C is okay because C is part of a candidate key (CD).

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

No, because B → C and B is not a candidate key (for either S1 and S2)

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

No, because BC is the intersection and BC is not a key for either S1 and S2

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

No, because CD→A (or A→C) are lost

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

Yes, A is a key for ABD and B is a key for BC (also all 2 attr relns are in BCNF)

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

Yes, Intersection is B and it is a key for BC

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

No, because CD→A (or A→C) are lost ;

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 (<u>TName</u>, TCity)

PLAYERS (<u>PName</u>, Team, Salary) [where Team is a foreign key referencing TEAMS]

GAMES (<u>HomeTm</u>, <u>AwayTm</u>, <u>Date</u>, 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".

$$\prod pname, salary(Players \bowtie ((\prod awayTm(\sigma_{HomeTm='Wombats' \wedge HomePts<AwayPts} G) \cup$$
$$(\prod HomeTm(\sigma_{AwayTm='Wombats \wedge AwayPts<HomePts} G))))$$

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

$\{T \mid \exists P \in Players(\exists G1 \in (G1.HomeTm = 'Wombats' \wedge G1.HomePts < G1.AwayPts$

$\wedge\ G1.AwayTm = P.Team$

$\wedge\ T.PName = P.Pname \wedge T.Salary = P.Salary))$

$\vee$

$(\exists G2 \in Games(G2.AwayTm = 'Wombats'$

$\wedge\ G2.HomePts > G2.AwayPts$

$\wedge\ G2.HomeTm = P.Team$

$\wedge\ T.Pname = P.Pname$

$\wedge\ T.Salary = P.Salary))\}$

Name:______________________________ SID:____________________

Question 3 (continued)

Recall the schema: **TEAMS (<u>TName</u>, TCity); PLAYERS (<u>PName</u>, Team, Salary)**

GAMES (<u>HomeTm</u>, <u>AwayTm</u>, <u>Date</u>, HomePts, AwayPts)

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

$$\prod TName(Teams) - \prod HomeTm(\sigma_{HomePts<AwayPts} G)$$

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

$$\{T \mid \forall G \in Games(T.TName = G.HomeTm \Rightarrow G.HomePts > G.AwayPts)\}$$

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

$$\{T \mid \forall T1 \in Teams(T1.TCity = 'Sydney' \Rightarrow$$
$$\exists G \in Games(G.AwayTm = T1.TName \wedge T.Name = G.HomeTm))\}$$

.

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

Consider the schema from question 3:

TEAMS (<u>TName</u>, TCity)

PLAYERS (<u>PName</u>, Team, Salary) [where Team is a foreign key referencing TEAMS]

GAMES (<u>HomeTm</u>, <u>AwayTm</u>, <u>Date</u>, 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.

SELECT Distinct P.PName, P.Team, P.Salary

FROM Players P, Games G

WHERE

G.HomeTm = P.Team

AND G.AwayTm = 'Koalas'

AND G.HomePts > G.AwayPts

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

SELECT T.TName

FROM Teams T

WHERE NOT EXISTS

(SELECT *

FROM Games G

WHERE G.HomeTm = T.Tname

AND G.AwayPts > G.HomePts)

Question 4 – SQL (continued)

Recall the schema: **TEAMS (TName, TCity); PLAYERS (PName, Team, Salary)**

GAMES (HomeTm, AwayTm, Date, HomePts, AwayPts)

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

```
SELECT Tname
FROM Teams T
WHERE NOT EXISTS((
SELECT TName
FROM Teams T2
WHERE T2.TCity = 'Sydney')
MINUS
(SELECT AwayTM
FROM Games G
WHERE G.HomeTm = T.Tname))
```

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.

```
SELECT T.TName, T.TCity, COUNT(*) as numPlayers, SUM(P.Salary) AS Payroll
FROM TeamsT, Players P
WHERE T.TName = P.Team
GROUP BY TName, TCity
HAVING numPlayers > 10
ORDER BY Payroll DESC
```

Question 4 – SQL (continued)

Recall the schema: **TEAMS (<u>TName</u>, TCity); PLAYERS (<u>PName</u>, Team, Salary)**

GAMES (<u>HomeTm</u>, <u>AwayTm</u>, <u>Date</u>, HomePts, AwayPts)

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).

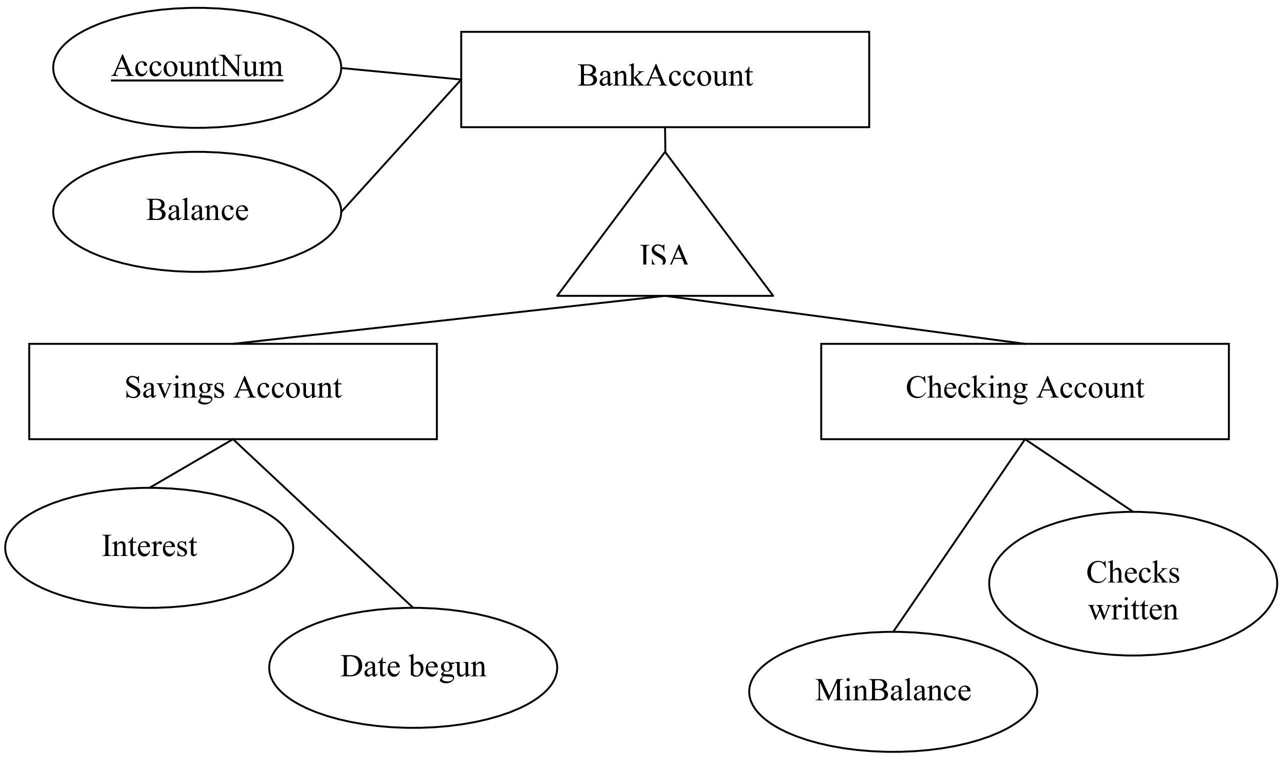
```
CREATE VIEW HomeWins as
SELECT HomeTm, COUNT(*) as numWins
FROM Games G
WHERE G.HomePts > G.AwayPts
GROUP BY HomeTm

SELECT HomeTm
FROM HomeWins H
WHERE H.numWins = (SELECT MAX(numWins) FROM HomeWins)
```

f) **[3 points]** Under what condition 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).

If there are teams without players.

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

b) [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.



c) [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.

1) Requires joins to look at all accounts.

CREATE TABLE BankAccount(

AccountNum INTEGER PRIMARY KEY,

Balance FLOAT);

CREATE TABLE SavingsAccount(

AccountNum INTEGER,

Interest FLOAT,

Date DATE,

PRIMARY KEY(AccountNum),

FOREIGN KEY (AccountNum) REFERENCES BankAccount);

CREATE TABLE CheckingAccount(

AccountNum INTEGER,

ChecksWritten INTEGER,

MinBalance FLOAT,

PRIMARY KEY(AccountNum),

FOREIGN KEY (AccountNum) REFERENCES BankAccount);

2) Requires selects to look at specific types of accounts.

CREATE TABLE Accounts(

AccountNum INTEGER PRIMARY KEY, Balance FLOAT,

Type INTEGER,

Interest FLOAT, Date DATE,

ChecksWritten INTEGER, MinBalance FLOAT);

Question 5 (continued)

d) [3 points] Briefly describe how relational "views" could help make query writing easier for the tables that result from one 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.

Easier to examine specific types of accounts by writing

CREATE VIEW SavingsAccounts AS

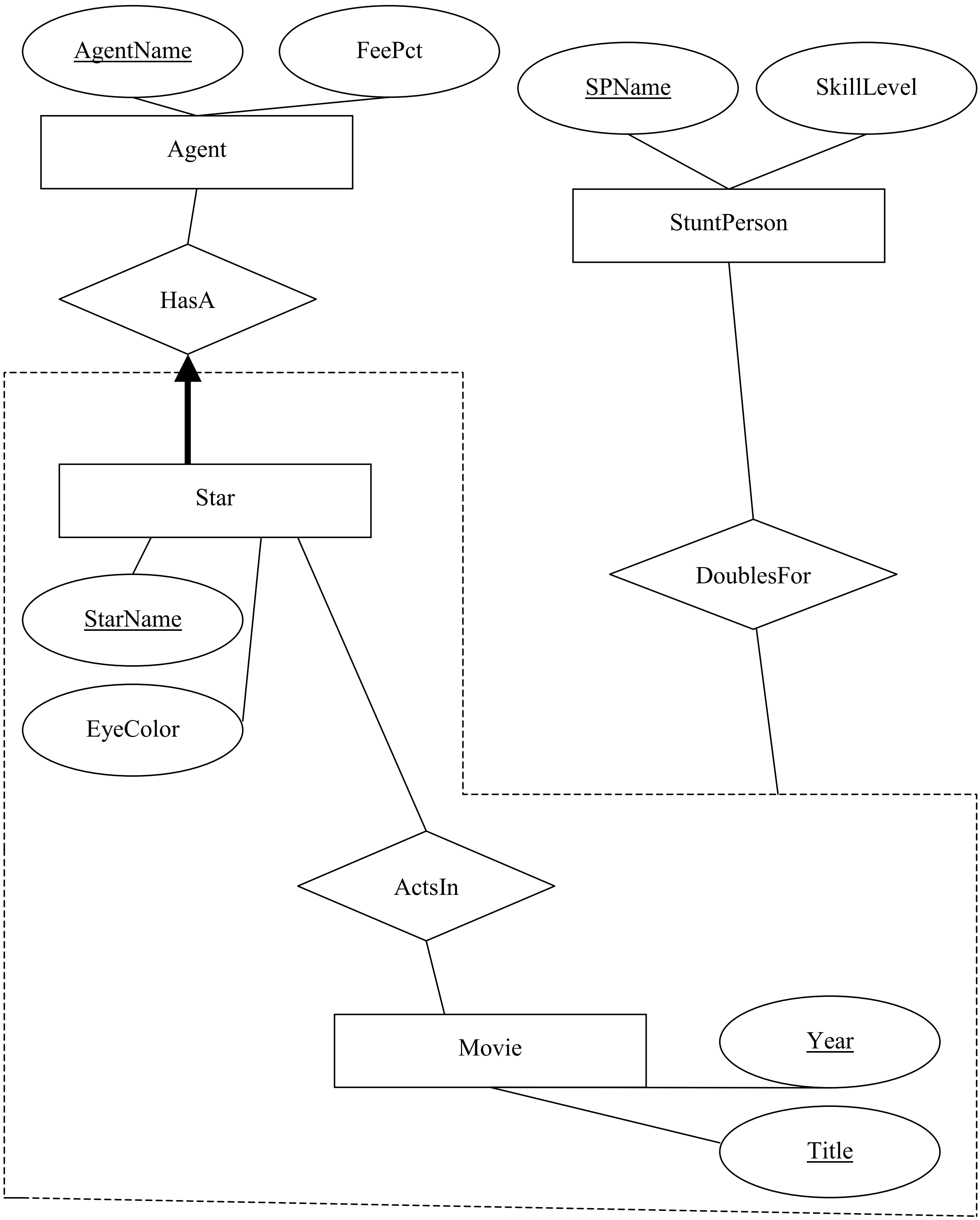
SELECT * FROM Accounts WHERE type = 1;

CREATE VIEW CheckingAccounts AS

SELECT * FROM Accounts WHERE type = 2;

e) [10 points] Draw the ER diagram that represents the schema on the next page. Be sure that all constraints, 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 eachother.***

Name:______________________________ SID:____________________


AgentName
FeePct
Agent
HasA
SPName
SkillLevel
StuntPerson
Star
StarName
EyeColor
DoublesFor
ActsIn
Movie
Year
Title

Question 5 (continued) (REMOVE THIS PAGE IF YOU LIKE)

```
CREATE TABLE Agent (
  AgentName varchar(50) PRIMARY KEY,
  FeePct float);

CREATE TABLE Star (
  StarName varchar(50) PRIMARY KEY,
  Eyecolor varchar(10)
   AgentName varchar(50)  NOT NULL,
  FOREIGN KEY (AgentName) REFERENCES Agent);

CREATE TABLE Movie (
  Title varchar(50),
  Year integer,
  PRIMARY KEY (Title, Year));

CREATE TABLE StuntPerson (
  SPName varchar(50) PRIMARY KEY,
  SkillLevel integer);

CREATE TABLE ActsIn (
  StarName varchar(50),
  Title varchar(50),
  Year integer,
  PRIMARY KEY (StarName, Title, Year),
  FOREIGN KEY (StarName) REFERENCES Star,
  FOREIGN KEY (Title, Year) REFERENCES Movie);

CREATE TABLE DoublesFor (
  SPName varchar(50),
  StarName varchar(50),
  Title varchar(50),
  Year integer,
  PRIMARY KEY (SPName, StarName, Title, Year),
  FOREIGN KEY (SPName) REFERENCES StuntPerson,
  FOREIGN KEY (StarName, Title, Year) REFERENCES ActsIn);
```