Answer Key

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

CS186
Fall 2003

Eben Haber
Midterm

Midterm Exam: Introduction to Database Systems

This exam has seven 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!*

Please **write your login** at the top of **every page;** 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!

I		/24
II		/10
III		/6
IV		/10
V		/10
VI		/20
VII		/20

Total		/100

I. Query Languages (30 points).

Given the following database tables, choose all queries in SQL, Relational Algebra or Tuple Relational Calculus that return the proper answer to the corresponding question. Note that each question may have more than one correct answers. Circle all that apply.

Primary keys are underlined. Note that the driver involved in a car accident may not always be the owner of the car. Assume that accident_date is of type integer, and represents a year (e.g. 1980). Year is also of type integer. We assume that a car cannot get involved in more than one accident at a certain date.

Person(SSN, name, address)
Car(license, year, model)
Accident(license, accident_date, driver, damage_amount)
Owns(SSN, license)

1. (4 points) Find the SSN of every person who owns one or more cars, none of which has ever been involved in a car accident.

   A. SELECT O.SSN
   FROM Owns O
   WHERE O.license NOT IN (SELECT A.license
   FROM Accident A)

   B. $\pi_{SSN}(Owns) - \pi_{SSN}(Owns \triangleright\!\triangleleft Accident)$

   C. $\{O \mid \exists O1 \in Owns(O.SSN = O1.SSN \wedge \neg\exists A \in Accident(O1.license = A.license))\}$

   D. None of the above

   B is right. A and C return the SSN of every person that has at least one car that is not involved in an accident, while the question asks for the people for whom all of their cars have never been involved in an accident.

2. (4 points) Find the SSN of every person, who owns a TOYOTA or a DODGE

   A. SELECT O.SSN
   FROM Owns O, Car C
   WHERE O.license=C.license AND
   (C.model='TOYOTA' OR C.model='DODGE')

   B. $\pi_{SSN}(Owns \triangleright\!\triangleleft \sigma_{model='TOYOTA' \vee model='DODGE'}(Car))$

   C. $\{O \mid \exists O1 \in Owns(O1.SSN = O.SSN \wedge$
   $\exists C \in Car(O1.license = C.license \wedge (C.model = 'TOYOTA' \vee C.model = 'DODGE')))\}$

   D. None of the above

   A, B and C are right.

3. (4 points) Find the SSN of all persons who have had all of their cars involved in an accident.

   A. SELECT O.SSN
   FROM Owns O, Accident A
   WHERE O.license=A.license
   EXCEPT
   SELECT O.SSN
   FROM Owns O
   WHERE O.license NOT IN (SELECT A.license
   FROM Accident A)

   B. $\pi_{SSN,license}(Owns \triangleright\triangleleft Accident) \div \pi_{license}(Accident)$

   C. $\{O \mid \exists O1 \in Owns(O1.SSN = O.SSN \wedge \forall O2 \in Owns(O1.SSN = O2.SSN \Rightarrow \exists A \in Accident(O2.license = A.license)))\}$

   D. None of the above.

   A and C are right. B returns the people who have been involved in every accident. To be right it should be $\pi_{SSN,license}(Owns \triangleright\triangleleft Accident) \div \pi_{license}(Owns)$

4. (4 points) Who is the driver who participated in the most costly accident? Return the driver and the amount of damage.

   A. SELECT driver, damage_amount
   FROM Accident
   WHERE damage_amount IN (SELECT MAX(damage_amount)
   FROM Accident)

   B. SELECT driver, damage_amount
   FROM Accident
   WHERE damage_amount=MAX(damage_amount)

   C. (SELECT driver, damage_amount FROM Accident)
   EXCEPT
   (SELECT a1.driver, a1.damage_amount
   FROM Accident a1, Accident a2
   WHERE a1.damage_amount<a2.damage_amount AND a1.driver<>a2.driver)

   D. None of the above.

   A is right. B doesn't work. C doesn't work if the driver involved in the most expensive accident, has also been involved to some other accident.

5. (4 points) Find the license number of all cars that had an accident in less than 5 years after their production. Return the license number and the number of years

   A. SELECT c.license
   FROM Car c, Accident a
   WHERE a.accident_date-c.year<5

   B. $\pi_{license}(\sigma_{accident\_date-year<5}(Car \triangleright\triangleleft Accident))$

C. $\{C \mid \exists C1 \in Car(C.license = C1.license \wedge \exists A \in Accident(C1.accident\_date - A.year < 5))\}$

D. None of the above

D is right. None of the queries return the number of years.

6. (4 points) Find the accident with the smallest cost and return the corresponding owner of the car, his/her address and the amount of damage.

A. SELECT O.SSN, P.address, MIN(A.damage_amount)
FROM Accident A, Owns O, Person P
WHERE O.license=A.license AND O.SSN=P.SSN
GROUP BY O.SSN, P.address

B. SELECT O.SSN, P.address, (SELECT MIN(damage_amount) FROM Accident)
FROM Owns O, Person P, Accident A
WHERE O.SSN=P.SSN AND A.license=O.license

C. SELECT O.SSN, P.address, A.damage_amount
FROM Owns O, Person P, Accident A
WHERE O.SSN=P.SSN AND A.damage_amount=(SELECT MIN(damage_amount)
FROM Accident)

D. None of the above.

The answer is D. A returns the cheaper accident for each owner, B returns every owner involved in an accident, along with the minimum value of damage, and C doesn't work because it doesn't perform the join between Accident and the other relations.

7. (4 points) Find the license number of all cars that have been involved in more than one accident. (DO NOT RETURN DUPLICATES)

A. SELECT A1.license
FROM Accident A1, Accident A2
WHERE A1.license=A2.license AND A1.accident_date<>a2.accident_date

B. SELECT DISTINCT A1.license
FROM Accident A1
WHERE A1.license IN (SELECT A2.license
FROM Accident A2
WHERE A1.accident_date<>A2.accident_date)

C. SELECT license
FROM Accident
GROUP BY license
HAVING COUNT(accident_date)>1

D. None of the above

B and C are right. A returns duplicates.

8. (2 points) Explain, in English, what the following Relational Calculus query expresses:

$$\left\{C \mid \exists C1 \in Car\left(C1.model = C.model \wedge \forall C2 \in Car\left(\begin{array}{l} C1.model = C2.model \Rightarrow \\ \exists A \in Accident(A.licence = C2.license) \end{array}\right)\right)\right\}$$

"Return the model of those cars, for which all cars of that model have been involved in an accident" (e.g. If every car of model 'TOYOTA' has been involved in an accident, the query must return the model 'TOYOTA'. If there is one 'TOYOTA' car not involved in an accident, 'TOYOTA' should NOT be returned by the query)

II. External Sorting (10 points)

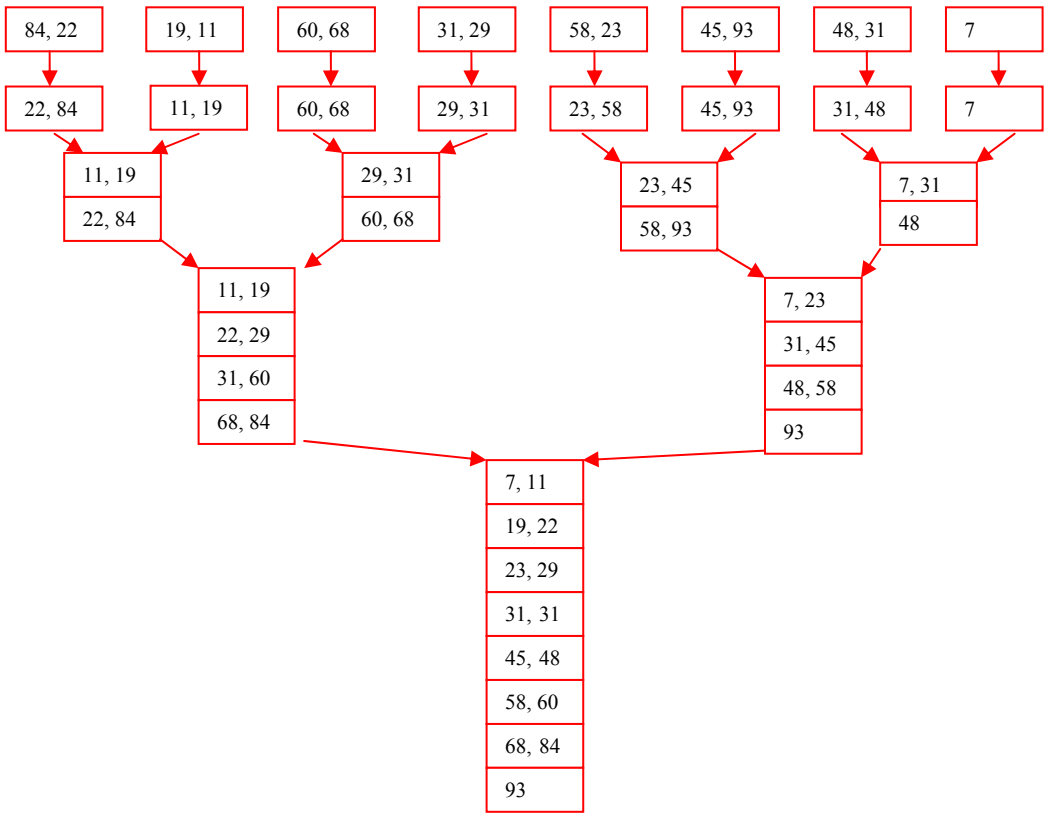
1. (6 points) Suppose we wish to sort the following values:

84, 22, 19, 11, 60, 68, 31, 29, 58, 23, 45, 93, 48, 31, 7

Assume that:

- you have three pages of memory for sorting
- you will use an external sorting algorithm with a 2-way merge
- a page only holds two values

For each sorting pass, show the contents of all temporary files.



2. (2 points) If you have 100 pages of data, and 10 pages of memory, what is the minimum number of passes required to sort the data.

Num passes = $1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil = 1 + \lceil \log_9 \lceil 100/10 \rceil \rceil = 1 + \lceil \log_9 10 \rceil = 1 + 2 = 3$

3. (2 points) If you have 500,000 pages of data, what is the minimum number of pages of memory required to sort the data in 3 passes?

Num passes = $1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil = 3$ so,

$\lceil \log_{B-1} \lceil N/B \rceil \rceil = 2 \Rightarrow \lceil N/B \rceil \leq (B-1)^2 \Rightarrow N \leq B(B-1)^2 \Rightarrow B = 81$

You can also simplify this to $N \leq B^3$, in which case B = 80.

III. Join Algorithms (6 points)

Consider a relation R with attributes (x,y) and a relation S with attributes (y, z). Column y in S is a key and the set of values of y in R are the same as the set of values of y in S. Assume that there are no indexes available and that there are 25 pages in the buffer available. Table R is 1,500 pages with 50 tuples per page. Table S is 400 with 100 tuples per page.

Compute the I/O costs for the following joins. Assume the simplest cost model, where pages are read and written one at a time.

A. Block nested loops join with R as the outer relation and S as the inner relation

$|R| + \lceil |R|/B \rceil \times |S| = 1500 + \lceil 1500/25 \rceil \times 400 = 25500$ or
$|R| + \lceil |R|/B\text{-}1 \rceil \times |S| = 1500 + \lceil 1500/24 \rceil \times 400 = 26700$ or
$|R| + \lceil |R|/B\text{-}2 \rceil \times |S| = 1500 + \lceil 1500/23 \rceil \times 400 = 27900$

Since all three equations were given in class, any were acceptable.

B. Block nested loops join with S as the outer relation and R as the inner relation

$|S| + \lceil |S|/B \rceil \times |R| = 400 + \lceil 400/25 \rceil \times 1500 = 24400$ or
$|S| + \lceil |S|/B\text{-}1 \rceil \times |R| = 400 + \lceil 400/24 \rceil \times 1500 = 25900$ or
$|S| + \lceil |S|/B\text{-}2 \rceil \times |R| = 400 + \lceil 400/23 \rceil \times 1500 = 27400$

C. Sort merge join with R as the outer relation and S as the inner relation

$2|R|(1 + \lceil \log_{B-1} \lceil |R|/B \rceil \rceil) + 2|S|(1 + \lceil \log_{B-1} \lceil |S|/B \rceil \rceil) + |R| + |S| =$
$2(1500)(1 + \log_{24} \lceil 1500/25 \rceil) + 2(400)(1 + \log_{24} \lceil 400/25 \rceil) + 1500 + 400 = 12500$

We cannot use the cost model of 3(M+N) because we cannot sort the larger of the two relations in 2 passes.

D. Sort merge join with S as the outer relation and R as the inner relation

Same as part (c)

E. Hash join with S as the outer relation and R as the inner relation

$3(|R| + |S|) = 3(1500 + 400) = 5700$

IV Query Optimization

Consider the following schema

Sailors(sid, sname, rating, age)
Reserves(sid, did, day)
Boats(bid, bname, size)

Reserves.sid is a foreign key to Sailors and Reserves.bid is a foreign key to Boats.bid.

We are given the following information about the database:
Reserves contains 10,000 records with 40 records per page.
Sailors contains 1000 records with 20 records per page.
Boats contains 100 records with 10 records per page.
There are 50 values for Reserves.bid.
There are 10 values for Sailors.rating (1...10)
There are 10 values for Boat.size
There are 500 values for Reserves.day

Consider the following query

```
SELECT S.sid, S.sname, B.bname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND
        B.size > 5 AND R.day = 'July 4, 2003';
```

(a) Assuming uniform distribution of values and column independence, estimate the number of tuples returned by this query.

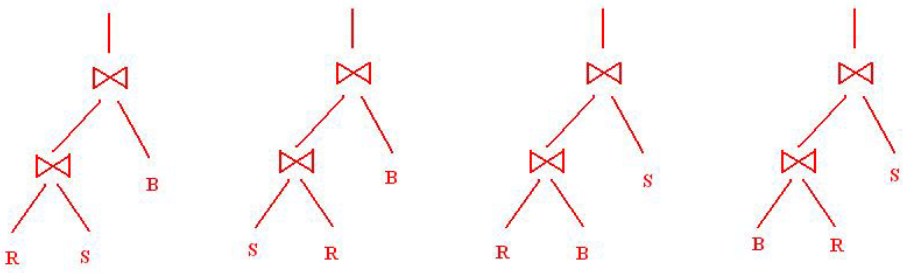
The maximum cardinality this query is $|R| \times |S| \times |B| = 10^9$.
Reduction Factors:
1/2 for B.size > 5,
1/500 for R.day = 'July 4, 2003',
1/100 for R.bid = B.bid, and
1/1000 for S.sid = R/sid

So number of tuples return is $(1/2)(1/500)(1/100)(1/1000)(10^9) = 10$

Consider the following query

```
SELECT S.sid, S.sname, B.bname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
```

(b) Draw all possible left-deep query plans for this query:



Note that we cannot join S and B since that would result in a cross product.

(c) For the first join in each query plan (the one at the bottom of the tree, what join algorithm would work best? Assume that you have 50 pages of memory. There are no indexes, so indexed nested loops is not an option.

$|R| = 250$, $|S| = 50$, $|B| = 10$.

R join S:

- Page-oriented nested loops join: $|R| + |R| \times |S| = 250 + 250 \times 50 = 12570$
- Block nested loops join: $|R| + \lceil |R|/49 \rceil \times |S| = 250 + 6 \times 50 = 550$
- Sort-merge join: note that $250 < (\text{num buffers})^2$ so cost is $3(|R| + |S|) = 3(250+50) = 900$
- Hash join: we can use hash join since each of the R partitions from phase one fit into memory. So cost is $3(|R| + |S|) = 3(250+50) = 900$.
- **Cheapest: Block nested loops**

S join R:

- Page-oriented nest loops join: $|S| + |S| \times |R| = 50 + 50 \times 250 = 12550$
- Block nested loops join: $|S| + \lceil |S|/49 \rceil \times |R| = 50 + 2 \times 250 = 550$
- Sort-merge join: $3(|R| + |S|) = 3(250+50) = 900$
- Hash join: $3(|R| + |S|) = 3(250+50) = 900$
- **Cheapest: Block nested loops**

B join R:

- Page-oriented nested loops: $|B| + |B| \times |R| = 10 + 10 \times 250 = 2510$
- Block nested loops join: $|B| + \lceil |B|/49 \rceil \times |R| = 10 + 250 = 260$
- Sort-merge join: $3(10+250) = 780$
- Hash join: $3(10+250) = 780$
- **Cheapest: Block nested loops**

R join B:

- Page-oriented nested loops join: The thing to note is that we can fit the entire Boats relation into memory, so we only need to scan Boats once. So the cost is $|R| + |B| = 260$. This is the same if we use simple nested loops join.
- Block nested loops join: $|R| + \lceil |R|/49 \rceil \times |B| = 250 + 6 \times 10 = 310$
- Sort merge join: $3(10+250) = 900$
- Hash join: $3(10+250) = 900$
- **Cheapest: Page-oriented nested loops, or simple nested loops**

V Buffer Management (10 points total)

In this exercise, you will compare the LRU, MRU, and CLOCK buffer management algorithms for a given workload (just as you did in the first part of homework 1). Here are the assumptions you must make:

- Assume there are four page slots your buffer manager must manage: P1, P2, P3, and P4.
- All four slots are empty to start.
- When the buffer pool has unused slots (such as at the beginning, when all four slots are empty), it will put newly read data in the leftmost empty slot (e.g., if slots 2 and 3 are free, it will use slot 2).
- The pages to be read from disk are labelled A through G.
- For each access the page is pinned, and then immediately unpinned.

Below are three tables for describing the contents of the buffer pool at each time step. A page is read at the beginning of each time step. You should record, in the table, the contents of each Buffer Page after the new page has been read in.

LRU (3 points)

Time	Page Read	Buffer Page			
		P1	P2	P3	P4
1	G	G			
2	F	G	F		
3	E	G	F	E	
4	D	G	F	E	D
5	G	G	F	E	D
6	C	G	C	E	D
7	F	G	C	F	D
8	E	G	C	F	E
9	D	D	C	F	E
10	C	D	C	F	E
11	D	D	C	F	E
12	B	D	C	B	E
13	A	D	C	B	A
14	D	D	C	B	A
15	F	D	F	B	A
Total number of pages "evicted":					7

MRU (3 points)

Time	Page Read	Buffer Page			
		P1	P2	P3	P4
1	G	G			
2	F	G	F		
3	E	G	F	E	
4	D	G	F	E	D
5	G	G	F	E	D
6	C	C	F	E	D
7	F	C	F	E	D
8	E	C	F	E	D
9	D	C	F	E	D
10	C	C	F	E	D
11	D	C	F	E	D
12	B	C	F	E	B
13	A	C	F	E	A
14	D	C	F	E	D
15	F	C	F	E	D
Total number of pages "evicted":					4

CLOCK (4 points)

Time	Page Read	Buffer Page			
		P1	P2	P3	P4
1	G	G			
2	F	G	F		
3	E	G	F	E	
4	D	G	F	E	D
5	G	G	F	E	D
6	C	C	F	E	D
7	F	C	F	E	D
8	E	C	F	E	D
9	D	C	F	E	D
10	C	C	F	E	D
11	D	C	F	E	D
12	B	C	B	E	D
13	A	C	B	A	D
14	D	C	B	A	D
15	F	F	B	A	D
Total number of pages "evicted":					4

VI. Functional Dependencies and Normalization (20 points total)

Consider the attributes A B C D E F G which have the following functional dependencies:

AD → F
AE → G
DF → BC
E → C
G → E

1. List all candidate keys (not superkeys): (5 points)

A D E

A D G

2. Which of the following dependencies are implied by those above: (5 points)
a. (IS) (IS NOT) G → C
b. (IS) (IS NOT) A → G
c. (IS) (IS NOT) ADF → E
d. (IS) (IS NOT) ADC → B
e. (IS) (IS NOT) AGF → E

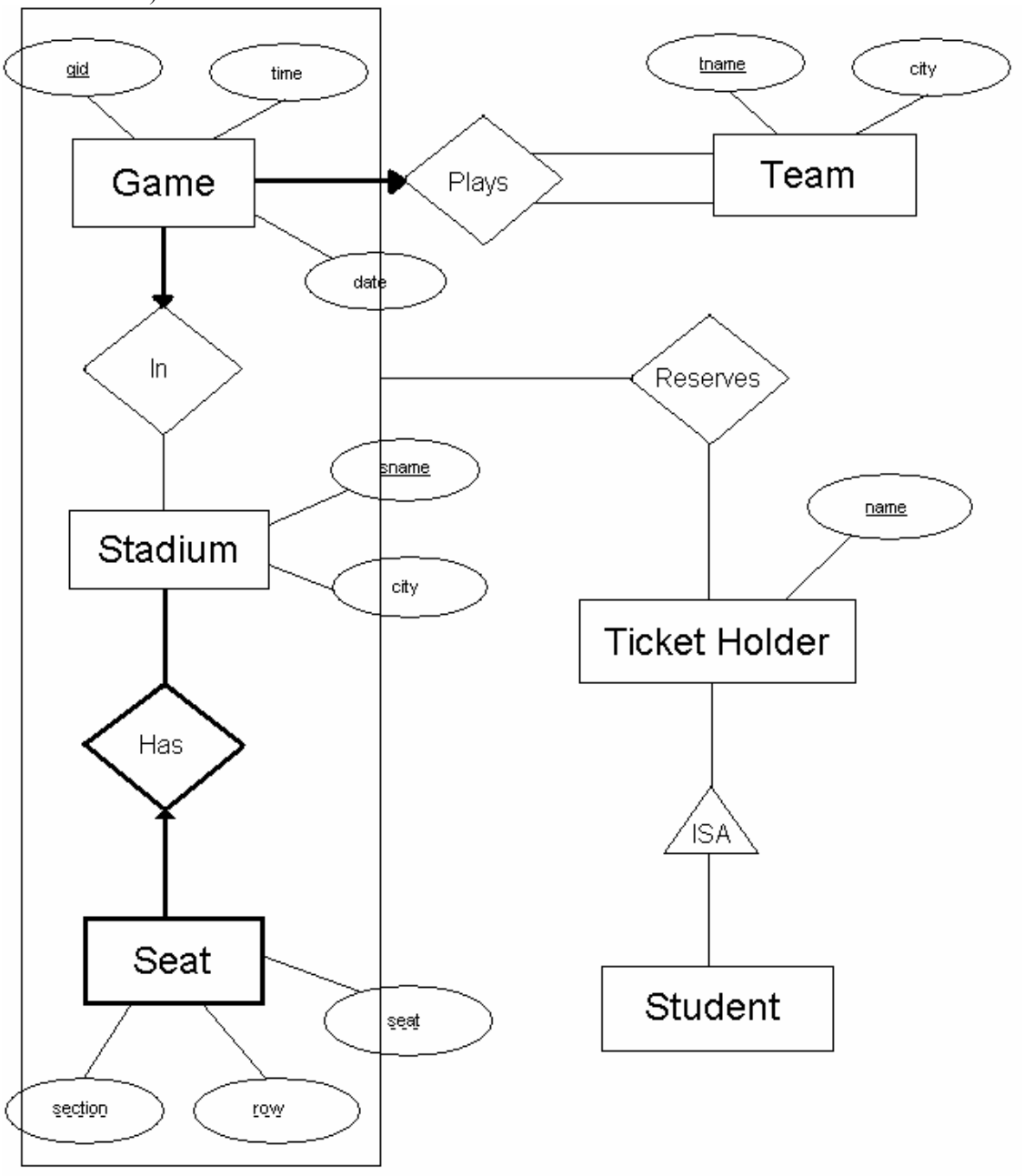
3. Consider the decomposition into 4 relations: (ADF) (CE) (EG) (ABDG).
This decomposition: (5 points)
a. (IS) (IS NOT) in BCNF
b. (IS) (IS NOT) in 3NF
c. (IS) (IS NOT) in 1NF
d. (IS) (IS NOT) Dependency Preserving
e. (IS) (IS NOT) Lossless

4. Consider the decomposition into 3 relations: (ADF) (EC) (ABDEG).
This decomposition: (5 points)
a. (IS) (IS NOT) in BCNF
b. (IS) (IS NOT) in 3NF
c. (IS) (IS NOT) 1NF
d. (IS) (IS NOT) Dependency Preserving
e. (IS) (IS NOT) Lossless

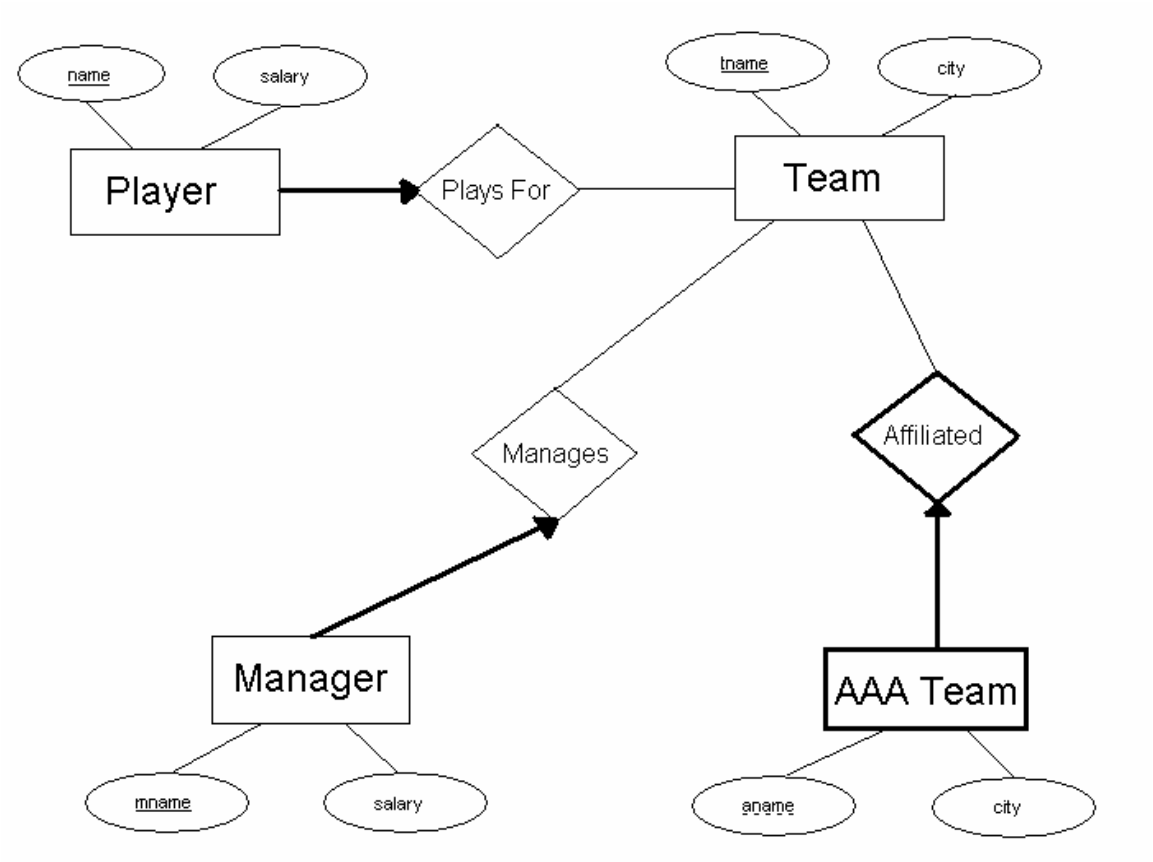
VII. Entity-Relational Model (20 Points)

a) (10 points) Draw an E-R diagram for the following situation:

- This is a simplified model for reserving baseball tickets.
- There are teams, which are identified by the team name. Teams are also located in a city.
- Teams play each other in games, which occur on a particular date at a particular time. Games are identified by a game ID, and each game has exactly two teams that play in it.
- A game is played in exactly one stadium.
- A stadium is identified by its name, and is also located in a city.
- Stadiums have seats, which have a section number, a row number, and a seat number.
- Ticket holders reserve seats for a game. Ticket holders are identified by their name.
- Some ticket holders are students (students get discounts, but we are not including that in the model).



b) (10 points) Using SQL, convert the following ER diagram to the relational model. Hint: You do not need CHECK constraints.




```
CREATE TABLE Player (name char(30), salary real, tname char(30) NOT NULL,
                     PRIMARY KEY (name), FOREIGN KEY (tname) REFERENCES Team);
CREATE TABLE Team (tname char(30), city char(30), PRIMARY KEY (tname));
CREATE TABLE Manager (mname char(30), salary real, tname char(30) NOT NULL,
                      PRIMARY KEY (mname),
                      FOREIGN KEY (tname) REFERENCES Team);
CREATE TABLE AAATeam (tname char(30), aname char(30), city char(30),
                      PRIMARY KEY (tname, aname),
                      FOREIGN KEY (tname) REFERENCES Team);
```