Database Management Systems

(COP 5725)

Spring 2019

Instructor: Dr. Markus Schneider

TA: Kyuseo Park

Exam 2 Solutions

Name:	
UFID:	
Email Address:	

Pledge (Must be signed according to UF Honor Code)

On my honor, I have neither given nor received unauthorized aid in doing this assignment.

_____

Signature

For scoring use only:

	Maximum	Received
Question 1	25	
Question 2	34	
Question 3	20	
Question 4	21	
Total	100	

Question 1 [25 points]

Consider the following schemas.

```
movie (mid, name, runtime, year, productionID)
production (pid, name)
actor (aid, name, year_of_birth, gender)
actin (mid, aid, salary, role)
```

1. [5 points] Write SQL commands to create tables for the schemas above. Assign meaningful SQL types to the attributes. Make sure to add meaningful and necessary integrity constraints.

```
create table movie
(mid varchar(20) not null,
 name varchar(40),
 runtime number,
 year number,
 productionID varchar(20)
 primary key(mid),
 foreign key(productionID) references Production(pid));

create table production
(pid varchar(20) not null,
 name varchar(40),
 primary key(pid));

create table actor
(aid varchar(20) not null,
 name varchar(40),
 year_of_birth number,
 gender char(2),
 primary key(aid));

create table actin
(mid varchar(20) not null,
 aid varchar(20) not null,
 salary number,
 role varchar(40),
 primary key(mid, aid)
 foreign key(mid) references movie(mid),
 foreign key(aid) references actor(aid));
```

2. [5 points] Insert the information provided in the paragraph below into the *movie* and *actin* tables. Assume that the value *mid* for the new movie is 'M010' and that the actor name *James* and the production name *Dijney* are unique.

   Production Dijney will release a new movie "Hello" in 2019 whose runtime is 90 minutes. Actor James who already acted in the movie "Hi" casted for the role "Hero1" and got $100,000.

```
insert into movie
select 'M010', 'Hello', 90, 2019, p.pid
from production p
where name = 'Dijney';

Alternatively:

insert into movie
values ('M010', 'Hello', 90, 2019,
        (select pid from production where name = 'Dijney'));

insert into actin
select 'M010', actor.aid, 100000, 'Hero1'
from actor
where name = 'James';

Alternatively:

insert into actin
values ('M010',
        (select aid from actor where name = 'James'),
        100000,
        'Hero1');
```

3. [3 points] Write a SQL command for creating an index on production names, and describe why an index is needed, in general.

```
create index production_name on Production(name);

An index is used to speed up searching on the indexed attributes in the
database.
```

4. [7 points] Write a SQL statement to increase the salary 10% for male actors and 20% for female actors, who acted in the movie "Stick".

```
update actin
set salary = case
               when aid in
                  (select ai.aid
                   from Actor a, ActIn ai, movie m
                   where m.mid = ai.mid and m.name = 'Stick' and
                         a.aid = ai.aid and a.gender = 'F')
                  then salary * 1.2
               when aid in
                   (select ai.aid
                    from Actor a, ActIn ai, movie m
                    where m.mid = ai.mid and m.name = 'Stick' and
                          a.aid = ai.aid and a.gender = 'M')
                    then salary * 1.1
             end;
```

5. [5 points] Write a SQL statement to delete all actors in the *actor* table who have never worked for the production "Dijney".

```
delete from Actor
where aid not in
          (select ai.aid
           from production p, movie m, actin ai
           where p.pid = m.productionID and m.mid = ai.mid and
                 p.name = 'Dijney');
```

Question 2 (SQL) [34 points]

Consider the following relational schema (primary keys are underlined)

branch (branch_name, branch_city, assets)

customer (customer_name, customer_street, customer_city)

account (account_number, branch_name, balance)

loan (loan_number, branch_name, amount)

depositor (customer_name, account_number)

borrower (customer_name, loan_number)

Write SQL statements to answer the following questions:

1. [7 points] Find the name of the branch with the largest number of customers (depositors and borrowers) in Gainesville.

```
select t1.branch_name, bnum + dnum as total
from
    (select branch_name, count (*) as bnum
     from branch natural join loan
     where branch_city = 'gainesville'
     group by branch_name) t1
    ,
    (select branch_name, count (*) as dnum
     from branch natural join account
     where branch_city = 'gainesville'
     group by branch_name) t2
     where t1.branch_name = t2.branch_name and
           bnum+dnum >= all
                        (select bnum + dnum from
                              (select branch_name, count (*) as bnum
                               from branch natural join loan
                               where branch_city = 'gainesville'
                               group by branch_name) t1
                              ,
                              (select branch_name, count (*) as dnum
                               from branch natural join account
                               where branch_city = 'gainesville'
                               group by branch_name) t2
                               where t1.branch_name = t2.branch_name
                        );
```

2. [5 points] Find the names of depositors who live in Gainesville and do not borrow a loan.

```
select d.customer_name
from depositor as d, customer as c
where d.customer_name = c.customer_name and
      c.customer_city = 'Gainesville' and
      not exists
          (select *
           from borrower as b
           where d.customer_name = b.customer_name
          );
```

3. [4 points] Find the names of customers who live in Gainesville and have at least two accounts.

```
select c.customer_name
from customer c
where c.customer_city = 'Gainesville'
and  2 <= (select count(d.account_number)
           from depositor d
           where d.customer_name = c.customer_name and
                 c.customer_city = 'Gainesville'
           group by c.customer_name
          );
```

4. [4 points] Find the name of the branch and its total account balance that has the lowest total account balance of customers in Gainesville.

```
select b.branch_name, sum(balance)
from branch b, account a
where b.branch_name = a.branch_name and
      b.branch_city = 'Gainesville'
group by b.branch_name
having sum(balance) <= all(select sum(balance)
                           from branch b, account a
                           where b.branch_name = a.branch_name
                           and b.branch_city = 'Gainesville'
                           group by b.branch_name
                          );
```

5. [3 points] Find the average balance of all customers who live in Gainesville and have more than three accounts.

```
select d.customer_name, avg(balance)
from depositor as d, account as a, customer as c
where d.account_number = a.account_number and
      d.customer_name = c.customer_name and
      c.customer_city = 'Gainesville'
group by d.customer_name
having count(distinct d.account_number) >= 3;
```

6. [7 points] Find the name of customers and their current personal property (account balance – loan amount) who live in Gainesville, deposit to the bank in Jacksonville, and borrow from the bank in Gainesville.

```
select c.customer_name, d.balance - b.amount as personal-property from
(select customer_name, balance from customer natural join depositor
natural join account natural join branch
 where branch_city = 'Jacksonville') d
 ,
(select customer_name, amount from customer natural join borrower natural
join loan natural join branch
 where branch_city = 'Gainesville') b
 ,
(select customer_name from customer where customer_city = 'Gainesville')
c
where d.customer_name = b.customer_name and c.customer_name =
d.customer_name
```

7. [4 points] Find the name of customers who have accounts in every branch in Gainesville.

```
select customer_name
from customer natural join depositor natural join
     account natural join branch
where branch_city = 'Gainesville'
group by customer_name, branch_city
having count (distinct(branch_name)) =
        (select count (*) from branch where branch_city = 'Gainesville');
```

Question 3 (SQL) [20 points]

Consider the following table schemas (primary keys are underlined):

```
member (m_number, name, age)
book (isbn, title, publisher, edition)
borrowed (m_number, isbn, date)
author (isbn, author)
```

Assume attribute *date* in the table *borrowed* has a numerical type.

Write SQL statements for the following queries:

1. [5 points] Find every member's name who borrowed all books that John Williams has borrowed. Do not include John Williams in your answer.

```
select name from member m
where not exists
      (
       (select isbn from member natural join borrowed
        where name = 'John Williams')
       minus
       (select isbn from member natural join borrowed
        where name = m.name)
      )
and name != 'John Williams';
```

2. [4 points] Find the name of the member who last borrowed the third edition of the book "DB" that was published by "ABC".

```
select name
from borrowed natural join book natural join member
where title = 'DB' and publisher = 'ABC' and edition = 3
and date >= all
            (select date
             from borrowed natural join book natural join member
             where title = 'DB' and publisher = 'ABC' and edition = 3);
```

3. [5 points] Find the title of the book along with the number of authors and edition that is written by the largest number of authors. (Assume a book can be written by multiple authors. Books with different editions are considered as different books.)

```
select title, edition, count(author) as num_of_author
from book natural join author
group by edition, title, publisher
having count(author) >= all (select count(author)
                             from book natural join author
                             group by edition, title, publisher
                            );
```

4. [3 points] Find the name of the member who is stored as the first borrower in the system. Find the member's name, title of the book and date.

```
select name, title, date
from member natural join borrowed natural join book
where date = (select min(date) from borrowed);
```

5. [3 points] Find the names of the members who borrowed any book written by Mano.

```
select distinct(m.name)
from member m, borrowed b, author a
where m.mnumber = b.mnumber and b.isbn = a.isbn
and b.isbn = some (select isbn from author where author = 'Mano');
```

Question 4 (QBE) [21 points]

Consider we have the following schema.

```
student (sid, sname, did, GPA)
took (sid, coid, score)
course-offering (coid, cid, semester, year, tid)
course(cid, cname, did)
teacher(tid, tname)
department(did, dname)
```

Draw Query-By-Example (QBE) tables for the following queries:

1. [4 points] Find the names of the courses that CISE department offers in 2019 but not in 2018.

course	cid	cname	did
	_CID	P.	_X

department	did	dname
	_X	CISE

Course-offering	coid	cid	semester	year	tid
		_CID		2019	
¬		_CID		2018	

2. [4 points] Find the names of the teachers who taught the courses CISE offered in 2018.

teacher	tid	tname
	_Z	P.

department	did	dname
	_X	CISE

Course-offering	coid	cid	semester	year	tid
		_Y		2018	_Z

course	cid	cname	did
	_Y		_X

3. [4 points] Find the names of all students who did not take any class in 2018.

student	sid	sname	did	GPA
	_X	P.		

took	sid	cid	score
¬	_X	_Y	

Course-offering	coid	cid	Semester	year	tid
		_Y		2018	

4. [5 points] Find the names of the teachers who have taught only once.

Course-offering	coid	cid	Semester	year	tid
					G._X

teacher	tid	tname
	_X	P.

conditions
CNT.All._X = 1

5. [4 points] Find the average GPA of all students who took the DB class in Fall 2018.

course	cid	cname	did
	_X	DB	

Course-offering	coid	cid	Semester	year	tid
		_X	Fall	2018	

took	sid	cid	score
	_Y	_X	

student	sid	sname	did	GPA
	_Y			P.AVG.ALL