Name:

Section (315/415):

MIDTERM EXAM - 600.315/415 - Databases

Date: October 27, 2011 in class

The total number of points in this exam is 72 for 600.315 and 75 for 600.415 students, although the difficulty of questions differ. If you work at approximately 1 minute per point, you should finish on time.

Question 1 - Relational Algebra (5 points)

Given the following relations r and s:

r

Age	IdNum
30	1
31	2
62	3
77	5
19	8

s

IdNum	Dept
2	600
3	520
6	050
6	150
7	600

compute the *full outer join* (**r** $\bowtie$ **s**):

Age	*IdNum*	*Dept*
30	*1*	*null*
31	*2*	*600*
62	*3*	*520*
77	*5*	*null*
null	*6*	*050*
null	*6*	*150*
null	*7*	*600*
19	*8*	*null*

Question 2 - Relational Algebra (5 points)

Express the following query in Relational Algebra. The tables that are used in Questions 2-9 are found on your supplementary handout.

- (5 points) List the names and phone numbers of all persons in the database who received parking tickets on 10-27-2011.

$CarTicket1027 \leftarrow \Pi_{LNO}(\sigma_{Date=10-27-2011}(TICKET))$

$Result \leftarrow \Pi_{Name,Phone}\ (PERSON \bowtie OWNS \bowtie CarTicket1027)$

Question 2 - Relational Algebra (continued)

(b) (5 points) List the name and SSN of all people (presumably collectors) who own every model of Porsche there is.

Porsches $\leftarrow \Pi_{make,model}\ (\sigma_{make=`Porsche`}\ (\text{CAR}))$

PersonPorsches $\leftarrow \Pi_{name,ssn,make,model}\ (\text{PERSON} \bowtie \text{OWNS} \bowtie \text{CAR})$

Result $\leftarrow \Pi_{name,ssn}\ (\text{PersonPorches} \div \text{Porsches})$

(c) (5 points) **SKIP**

(d) (5 points) **SKIP**

Question 3 - Tuple Relational Calculus (5 points)

Express the following query in Tuple Relational Calculus:

- (5 points) Print the license number of all red cars made in 2011 (year=2011) that are owned by someone who owns a car made before 1955.

$$\{ \text{t} \mid \exists c1 \in CAR(t[LNO] = c1[LNO] \wedge c1[Year] = 2011 \wedge \text{c1[color]= red}) \wedge (\exists o1 \in OWNS(o1[LNO] = c1[LNO] \wedge (\exists c2 \in CAR(c2[Year] < 1955 \wedge (\exists o2 \in OWNS(c2[LNO] = o2[LNO] \wedge o1[SSN] = o2[SSN])))))\}$$

Question 4 - SQL (10 points)

Express the following query in SQL.

(a) (5 points) List the names and SSN's of all people in the database who **don't** own car with a make of Toyota (*make='Toyota'*) .

```
SELECT DISTINCT Person.SSN, Person.Name
FROM Person
WHERE Person.SSN NOT IN ( SELECT Owns.SSN
                          FROM Car,Owns
                          WHERE Car.LNO = Owns.LNO AND Car.Name = "Toyota")
```

(b) (5 points) List the names and phone numbers of people in the database along with the total number of parking tickets each has received.

If the person has received no tickets, it is not necessary to list them.

```
SELECT   P.Name, P.Phone, COUNT(*)
FROM     Person P, Owns O, Ticket T
WHERE    P.SSN = O.SSN
   AND   O.LNO = T.LNO
GROUP BY P.SSN, P.Name, P.Phone
```

Question 5 - SQL (6-12 points)

Express the following queries in SQL.

(a) (* 6 points - *600.415 only*) Print the name and SSN of the person who has received the *most* number of parking tickets. You can assume that this person is unique.

```
SELECT   P.Name, P.SSN
FROM     Person P, Owns O, Ticket T
WHERE    P.SSN=O.SSN  AND  O.LNO=T.LNO
GROUP BY P.SSN
HAVING   COUNT(T.TNO) >= ALL ( SELECT   COUNT(T.TNO)
                               FROM     Person P1, Owns O1, Ticket T1
                               WHERE    P1.SSN=O1.SSN AND O1.LNO=T1.LNO
                               GROUP BY P1.SSN )
```

(b) (6 points-600.315 **and** 415) List the name of all people in the database who *don't* own a car that is the same color as a car that Aaron Smith owns.

(If Aaron Smith owns cars that are red and blue, list people who don't own a car that is either red or blue.)

```
SELECT P1.Name
FROM   Person P1
WHERE  P1.SSN NOT IN ( SELECT P.SSN
                       FROM   Person P, Person Psmith, Owns O, Owns Osmith,
                                  Car C, Car Csmith
                       WHERE  Psmith.Name = 'Aaron Smith'
                       AND    Psmith.SSN  = Osmith.SSN
                       AND    Osmith.LNO  = Csmith.LNO
                       AND    Csmith.Color = C.Color
                       AND    C.LNO = O.LNO
                       AND    O.SSN = P.SSN )
```

Question 6 - QBE (15 points)

Express the following queries in QBE. To simplify your work, table shells have been provided. Just fill in the appropriate cells with variables/values.

(a) (5 points) Print the SSN, Name and home zip code of all people who have received parking tickets in the same zip code as their home.

PERSON	SSN	Name	Phone	HomeZip
	P._s	**P.**		**P._z**

CAR	LNO	Color	Make	Model	Year

OWNS	SSN	LNO
	_s	**_ln**

TICKET	TNO	LNO	Date	TicketIssuedZip
		_ln		**_z**

(b) (5 points) Print the name everyone who owns the same color, model **and** make of car as one of the cars that Aaron Smith owns.

(For example, if Aaron owns a red Honda Accord, then find *all* owners of red Honda Accords. Aaron may own more than one car.)

PERSON	SSN	Name	Phone	HomeZip
	_asmith **_othssn**	**Aaron Smith** **P.**		

CAR	LNO	Color	Make	Model	Year
	_asmithlno **_othlno**	**_ascolor** **_ascolor**	**_asmake** **_asmake**	**_asmodel** **_asmodel**	

OWNS	SSN	LNO
	_asmith **_othssn**	**_asmithlno** **_othlno**

TICKET	TNO	LNO	Date	TicketIssuedZip

(c) (*5 points) List the zipcodes of all stores that have more than 1 'spark plug' manufactured by 'Acme Auto Parts' in stock, *or* have performed a repair

MANUFACTURER	MID	Name	Street_Address	City	State	Zipcode
	_m	**Acme Auto Parts**				

PART	MID	Serial_Num	Name	Retail_Cost	Wholesale_Cost	Shipping_Days
	_m	**_sn**	**Spark Plug**			

STORE	Store_Id	Street_Address	City	State	Zipcode
	_s1				**P.**
	_s2				**P.**

STOCK	MID	Serial_Num	Store_Id	Number_In_Stock
	_m	**_sn**	**_s1**	**> 1**

REPAIR	Repair_Id	Repair_Name	Labor_Cost

REPAIR_PARTS_REQUIRED	Repair_Id	MID	Serial_Num	Quantity_Required
	_r	**_m**	**_sn**	

JOB	Job_Number	Customer_Name	Date	Store_Id
	_j			**_s2**

JOB_REPAIRS	Job_Number	Repair_Id	Quantity_Of_ Repair
	_j	**_r**	

Question 7 - Understanding SQL Queries (10 points)

(a) (5 points - 600.315 only) Convert the following SQL statement to its English equivalent.

```
SELECT  Name
FROM    Person, Owns, Car
WHERE   person.SSN = owns.SSN
  AND   owns.LNO = car.LNO
  AND   car.Make = 'Honda'
  AND   car.Model = 'Accord'
  AND   car.Year = 2011;
```

Print the names of all people who own a 2011 Honda Accord.

(b) (* 5 points) Convert the following SQL statement to its English equivalent *(600.415 only)*.

```
SELECT Name
FROM   Person, Owns o1, Owns o2
WHERE  Person.SSN = o1.SSN
  AND  o1.SSN = o2.SSN
  AND  o1.LNO <> o2.LNO;
```

Print the names of all people who own more than 1 car.

Question 8 - Functional Dependencies (9 points)

Consider the relation NADDR(Name,Street,City,State,Zip), which you may represent as NADDR(N,S,C,T,Z). The Name attribute (N) is unique (and the chosen primary key). For any given zipcode, there is just one city and state. Also, for any given street, city and state, there is just one zipcode.

(a) Give an irreducible set of FDs for NADDR(N,S,C,T,Z) corresponding to the above restrictions.

$N \rightarrow SCTZ$

$Z \rightarrow CT$

$SCT \rightarrow Z$

(b) Are there any candidate keys besides N?

No. For an attribute to be a candidate key it must functionally determine all the attributes of a relation. As clearly multiple people may live on a given street, city, state and zipcode, none of these uniquely determine a name (N) in the general case (although there may be individual exceptions) so any candidate key must contain N, and since N itself is a candidate key, any superkey containing additional attributes beyond N is not minimal, and hence not a candidate key.

Alternately, one may use the FDs in part (a) above to compute that the closures $Z \rightarrow CTZ$ *and* $SCT \rightarrow SCTZ$ *do not functionally determine the entire relation, and no other attributes functionally determine more than themselves. So there are no other candidate keys besides N.*

Question 9 - Functional Dependencies (6 points)

Consider the relational scheme R=(A,B,C), and the set F of functional dependencies:

A $\rightarrow$ BC

B $\rightarrow$ C

A $\rightarrow$ B

AB $\rightarrow$ C

Compute the canonical cover F_c of R (and show your work):

- *$A \rightarrow BC$ and $A \rightarrow B$ can be combined into $A \rightarrow BC$.*
- *A is extraneous in $AB \rightarrow C$ because F logically implies (F - $AB \rightarrow C$) $\cup$ $B \rightarrow C$. This assertion is true because $B \rightarrow C$ is already in our set of FDs.*
- *C is extraneous in $A \rightarrow BC$, since $A \rightarrow BC$ is logically implied by $A \rightarrow B$ and $B \rightarrow C$.*

Thus, our canonical cover is $A \rightarrow B$ and $B \rightarrow C$.