University of Waterloo: Midterm Examination **Winter Term 2017**

Student Name: ______________________________	
Student Signature: ______________________________	
Student Identification Number: ______________________________	

E&CE 356	Database Systems
Section: 001	Instructor: Paul Ward
Monday, February 13th 2017	7:00 PM to 8:30 PM
Duration of Exam:	1.5 hours
Number of Exam Pages (including cover sheet):	10 pages / 4 questions
Exam Type:	Closed Book

General Notes

1. No electronic devices including no calculators.
2. Read the entire questions *carefully* before answering.
3. State any assumptions clearly and whenever needed.
4. No questions permitted. When in doubt, make an assumption, write it down.
5. Some SQL reminders are listed on Page 10.
6. *aquila non captat muscas*

Question	**Maximum**	**Score**						**Total**
1	30	(a)	(b)	(c)	(d)	(e)	(f)	
2	30	(a)	(b)	(c)	(d)	(e)		
3	30	(a)	(b)	(c)		(d)		
3	10	(a)			(b)			
Total	100							

Database for the midterm

The following schema and data will be used for the questions on this midterm. It is a small database for a vet, providing details about pet owners, which pets they own, and some details about the pets. The table name (relation) is in **bold** and the primary key is underlined.

Owner

OwnerID int	Name varchar(10)	Phone char(8)
123	Bart	123-4567
456	Lisa	789-1011
789	Maggie	722-2222
101	Homer	123-4567
112	Maggie	123-4567

Owns

OwnerID int	PetID int	Registered date
123	1	11/11/2011
123	4	14/12/2013
456	2	21/03/2009
789	3	02/07/2015
123	5	NULL

Pet

PetID int	Name varchar(10)	Type varchar(10)	SubType varchar(10)	BirthDate date	Weight int
1	Rainbow	Dog	Beagle	2011-10-10	35
2	Loki	Cat	Siamese	2013-08-17	12
3	Smeagol	Dog	Labrador	2014-03-01	53
4	Ploppy	Dog	Poodle	2012-11-24	13
5	Reggie	Bird	Parrot	2008-05-11	1

PetTypes

Type varchar(10)
Dog
Cat
Bird

The relation "**Owner**" has OwnerID as its primary key and no foreign keys.

The relation "**PetType**" has a single column, Type, which is its primary key, and identifies all possible pet Types.

The relation "**Pet**" has PetID has its primary key. The attribute "Type" is a foreign key that references relation "**PetType**", attribute "Type".

The relation "**Owns**" has PetID as its primary key. The "PetID" is a foreign key, referencing relation **Pet**, attribute "PetID". The relation also records when a pet was first registered with the vet.

If it helps, the specific table creation commands for these relations as SQL tables are:

```
create table PetType(Type char(6), primary key(Type));

create table Owner(OwnerID int primary key, Name varchar(10), Phone varchar(10));

create table Pet(PetID int primary key, Name varchar(10), Type varchar(10),
                 SubType varchar(10), BirthDate Date, Weight int,
               foreign key(Type) references PetType(Type));

create table Owns(OwnerID int, PetID int primary key, Registered Date,
                  foreign key(OwnerID) references Owner(OwnerID),
                  foreign key(PetID) references Pet(PetID));
```

1. [30 marks] SQL Comprehension: Considering the database schema and the particular data on page 2, for each of the following queries identify
(i) in a single sentence, what is the query computing?
(ii) what is the output for this particular dataset?

(a) `select Name,Weight from Pet where Type='Dog' order by name;`

(b) `select Type,count(distinct PetID) as Num from Pet group by Type;`

(c) `select count(PetID)/count(distinct OwnerID) as AO from Owns;`

(d)
```
select O.Name,sum(Weight) as TPW from Owner as O natural join Owns
          inner join Pet as P using (PetID) group by O.Name order by TPW desc;
```

(e)
```
select O.Name,P.Name,P.BirthDate from Owner as O inner join Owns using (OwnerID)
         inner join Pet as P using (PetID) where Type='Dog' and not exists
          (select * from Pet as Q where P.BirthDate > Q.BirthDate and Type = 'Dog');
```

(f)
```
select Type, AvgWt from (select Type,A.AW,avg(weight) as AvgWt from Pet,
(select avg(weight) as AW from Pet) as A group by Type,A.AW having AvgWt < A.AW) as B;
```

2. [30 marks] Query Creation: Considering the database schema and the particular data on page 2, solve the following queries with *either* Relational Algebra *or* SQL.

(a) What is the average weight of dogs?

(b) What is the average weight of each Type of Pet, ordered by increasing weight?

(c) Who are the owners of the heaviest pets, by Type? You should list the owner's Name and the pet's Type.

(d) What is the average number of people per pet in each household? A household is determined by the phone number: two people (Owners) with the same phone number are deemed to be in the same household; those with different phone numbers are deemed to be in different households.

(e) What are the names and phone numbers of pet owners who own every type of pet?

3. [30 marks] Normalization: Considering the database schema on page 2, in addition to the functional dependencies implied by the primary keys, the following functional dependency exists: SubType $\rightarrow$ Type

(a) What relations, if any, need to be changed to make the schema Third-Normal Form? If changes are required, identify them, including any new relations, and any changes to primary and foreign keys.

(b) After your adjustments from part(a), if any, is the schema now also in BCNF? If any further changes are required, identify them, including any new or changed relations, and any changes to primary and foreign keys.

(c) We would like the schema to (i) allow owners to have more than one phone number (ii) allow pets to be owned by more than one owner. If the current schema does not support this, identify why not.

(d) If needed, adjust the schema to support the requirements from part(a). If you need to make adjustments, identify any new or changed relations, and any changes to primary and foreign keys.

4. [**10 marks**] **Debugging:** Considering the database schema on page 2, some queries have been created that are not correct.

(a) We would like the names of all dog owners and so we execute the query:

```
select name from Owner natural join Owns natural join Pet where Type='Dog';
```

What is wrong with this query?

(b) Sometimes pets die and are removed from the Pet relation. This may be the last pet that an Owner had with us. We would like to identify any owners who have no Pets and so we use the query:

```
select name, Phone from Owner where name not in
        (select name from Owner inner join Owns using (OwnerID));
```

What is wrong with this query?

Some SQL Reminders

DDL:

```
create table
drop table [if exists <table>]
alter table <table> add <attribute> <domain>
alter table <table> drop <attribute>
create view <view> as <query expression>

Domain Types: char(n), varchar(n), int, smallint, numeric(p,d)
              real, double precision, float(n), date, time

Integrity     not null, default <V>, primary key (<a1, ..., an>),
Constraints:  unique(<a1, ..., an>),
              foreign key (<am, ..., an>) references (<am’, ..., an’>)

Referential   on update <...>, on delete <...>
Actions:      cascade, set null, set default, no action

Check Constraints: check (<predicate>)
```

DML:

```
insert into <table> values (...)
update <table> set <attribute = ...> where <predicate>
delete from <table> where <predicate>
select <a1,...,an> from <t1,...,tn> where <predicate>

Options on selection attributes: distinct, as

Options on selection tables: natural join, inner join, outer join
                             <join type> on <predicate> using <attributes>

Set Operations: union, interset, except (use ‘all’ to retain duplicates)

Aggregate Functions: avg, min, max, sum, count

Grouping: group by, having, order by

Options on predicate: =, !=, <, >, <=, >=, in, not in, exists, not exists,
                      is null, is not null, between, like <string pattern>
```