CS 500, Database Theory

Midterm Exam

Completed between July 24th (9am) and July 28 (5pm)

SOLUTIONS

Problem 1 (15 points) – True or False?

Answer the *true* or *false* questions below, providing a *very brief* explanation of your answers. You will not receive full credit without an explanation.

(a) (5 points) There exist one-to-one binary relationship sets in which *no key constraints* hold. True or false?

False. By definition of a one-to-one binary relationship set, each entity on one side is related to at most one entity on the other side. So, in a one-to-one binary relationship set, a key constraint holds on each participating entity set.

(b) (5 points) Suppose you are given an instance of a relation. By looking at this instance you can *know for sure* what the candidate keys of this relation are. True or false?

False. Candidate keys are derived from business rules. These must hold over all valid instances of a relation, not just one particular instance at hand. By looking at an instance we can see that some candidate keys don't hold (e.g., if we find duplicates in a column designated as a candidate key). But we cannot know for sure which keys do hold.

(c) (5 points) Consider the relation below. This relation *does not* have any superkeys. True of false?

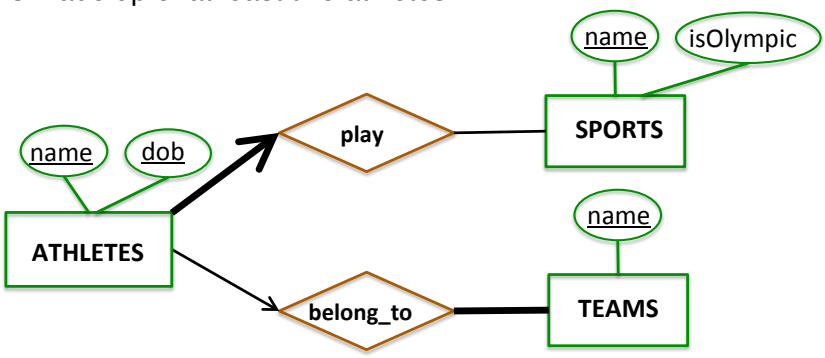
```
create table Foo (
  X          number,
  Y          number,
  Z          number not null,
  primary key (X, Y),
  unique (Z)
);
```

False. The relation Foo has the following superkeys: {X, Z}, {Y, Z} and {X,Y,Z}.

Problem 2 (20 points)

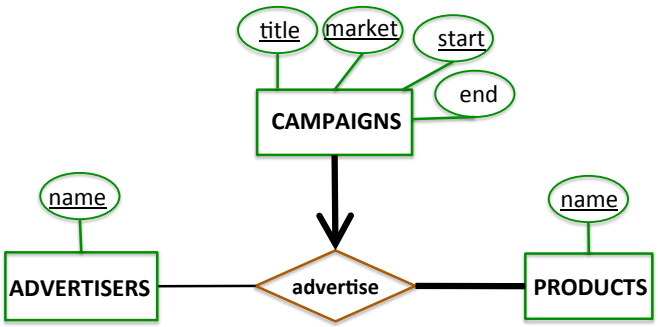
(a) (10 points) Consider the business rules below, describing a database of athletes, sports and teams. Draw an ER diagram that encodes these business rules. *Clearly mark all key and participation constraints*.

- An athlete is described by a name and a date of birth (dob), and no two athletes have the same combination of name and dob.
- A team is described by its name, and no two teams have the same name.
- A sport is described by its name and contains information about whether this is an Olympic sport or not. All sports have different names.
- Each athlete plays exactly one sport, and belongs to at most one team.
- Each team is made up of at least two athletes.



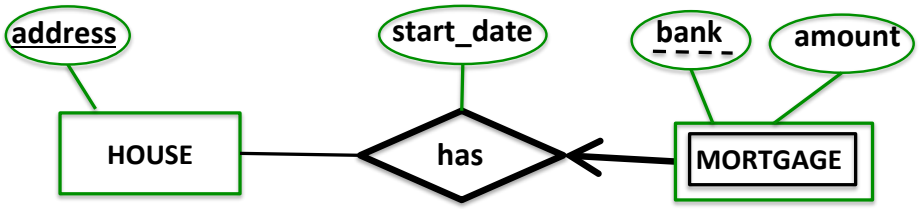
(b) (10 points) Consider the business rules below, describing a database of advertisers, campaigns and products. Draw an ER diagram that encodes these business rules. *Clearly mark all key and participation constraints*.

- An advertiser is described by its name. No two advertisers have the same name.
- An advertising campaign is described by campaign title, target market, and start and end dates. (Call these attributes title, market, start and end.) No two campaigns have the same combination of title, target market and start date.
- A product is described by its name. No two products have the same name.
- Each campaign advertises a particular product sold by a particular advertiser.
- Each product is advertised at least once. Products that are not part of any advertising campaigns by any advertisers are not stored in our database.
- Each campaign is associated with exactly one advertiser and exactly one product.



Problem 3 (15 points)

(a) (5 points) Consider an ER diagram below. Write SQL statements (create table) that implement the constraints specified by this ER diagram. Create as many tables as is required. Briefly explain which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your description. *You will not receive full credit without a proper description.*



(the line from MORTGAGE to has is bold and has an arrowhead)

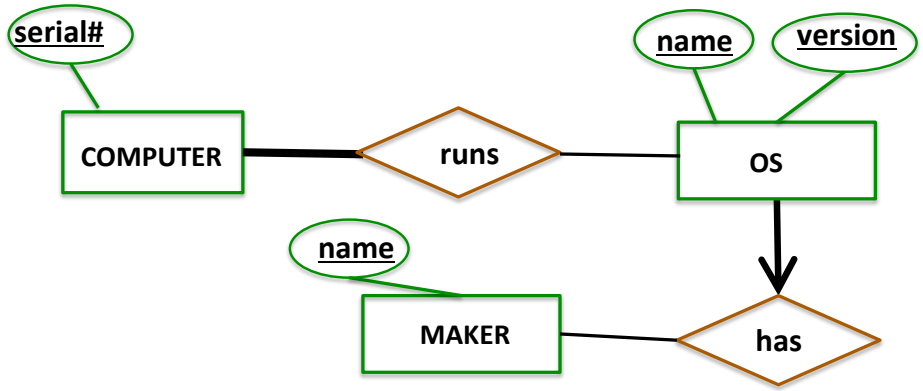
Solution

```
create table House (
  address    varchar(128) primary key
);

create table Mortgage (
  address    varchar(128),
  bank       varchar(128),
  amount      number,
  start_date date,
  primary key (address, bank),
  foreign key (address) references House(address)
            on delete cascade
);
```

This representation captures that Mortgage is a weak entity set, with House as its identifying entity set. This is captured by making address part of the primary key on Mortgage, and by specifying that the tuples in Mortgage should be deleted if the corresponding tuple in House is deleted.

(b) (10 points). Consider an ER diagram below. Write SQL statements (create table) that implement the constraints specified by this ER diagram. Create as many tables as is required. Briefly explain which constraints are captured in your relational implementation, and in what way. If a constraint cannot be implemented, state that explicitly in your description. *You will not receive full credit without a proper description.*



(the line from COMPUTER to runs is bold; the line from OS to has is bold and has an arrowhead; the other two lines are not bold and don't have arrowheads)

Solution

```
create table Computer (
  serial#    varchar(128) primary key
);

create table Maker (
 name  varchar(128) primary key
);

create table OS_Has_Maker (
  name  varchar(64),
  version varchar(32),
  maker varchar(128) not null,
  primary key (name, version),
  foreign key (maker) references Maker(name)
);

create table Runs (
 serial#     varchar(128),
 os_name     varchar(64),
 os_version  varchar(32),
 primary key (serial#, os_name, os_version),
 foreign key (serial#) references Computer(serial#),
 foreign key (os_name, os_version) references
                        OS_Has_Maker (name, version)
);
```

This implementation does not capture the participation constraint on Computer for the runs relationship set. Participation without a key constraint cannot be captured. Key constraint on OS for the has relationship is represented by there being a single table for OS and Maker (OS_Has_Maker), with OS name and version as the primary key. Participation constraint on OS for the has relationship set is represented by the not null constraint over maker in OS_Has_Maker, which ensures that each OS is associated with a maker value. This value then references Maker(name), correctly enforcing referential integrity.

Problem 4 (20 points)

Consider two relation instances below, with the given schemas. In each question below, *write a **relational algebra** expression* and ***show its result*** when the expression is executed with the given instances.

```
Performers (name, genre)
Grammy_Nominations (name, category, year, is_winner)
```

Performers

name	genre
Coldplay	Rock
Amy Winehouse	R&B
Arizona Shakes	Rock
Frank Ocean	R&B
The Black Keys	Rock

Grammy_Nominations

name	category	year	is_winner
Arizona Shakes	New Artist	2013	no
Arizona Shakes	Rock Performance	2013	no
Frank Ocean	Album of the Year	2013	no
Frank Ocean	Rap/Sung Collaboration	2013	yes
Coldplay	Rock Performance	2013	no
Coldplay	Rock Album	2009	yes
Coldplay	Album of the Year	2009	no
Coldplay	Rock Performance	2012	no
The Black Keys	Rock Performance	2013	yes

(a) (10 points) List the years in which a Rock performer won a Grammy, along with the name of the performer.

$$\pi_{year,\,name}(\sigma_{genre='Rock'}Performers) \rhd\lhd (\sigma_{is\_winner='Yes'}Grammy\_Noms))$$

year	name
2009	Coldplay
2013	The Black Keys

(b) (10 points) List years in which Coldplay was nominated for a Grammy but did not win.

$$\pi_{year}(\sigma_{name='Coldplay'}(Grammy\_Nominations) \setminus \sigma_{name='Coldplay'\ AND\ is\_winner='yes'}(Grammy\_Nominations))$$

year
2013
2012

Problem 5 (30 points) - SQL

Refer to the Performers database defined in Problem 4. In each question, ***write a SQL query*** and ***show its result*** when the expression is executed with the given instances.

(a) (10 points) List the names and genres of all performers who were nominated for a Grammy in 2013 and in 2009.

```
select P.name, P.genre
from Performers P, Grammy_Nominations N
where P.name = N.name
and N.year = 2013
INTERSECT
select P.name, P.genre
from Performers P, Grammy_Nominations N
where P.name = N.name
and N.year = 2009
```

A self-join query also works, with Grammy_Nominations being used twice.

name	**genre**
Coldplay	Rock

(b) (10 points) List the names and nomination years for all performers that were nominated for at least 2 Grammys in a given year.

```
select name, year
from Grammy_Nominations
group by name, year
having count(*) > 1
```

name	**year**
Arizona Shakes	2013
Frank Ocean	2013
Coldplay	2009

(c) (10 points) List the names of performers who were nominated for a Grammy at least once but never won.

```
select P.name, P.genre
from Performers P, Grammy_Nominations N
MINUS
select P.name, P.genre
from Performers P, Grammy_Nominations N
where N.is_winner = 'yes'
```

A not-exists query also works, but this one is more straight-forward.

name
Arizona Shakes