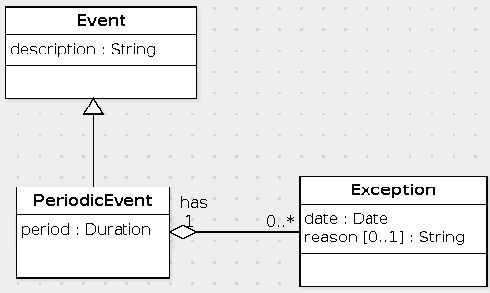
**CS 5200 Database Management Systems Mid-Term Exam Fall 2010**

This is a one-hour exam and there are 60 points in all.

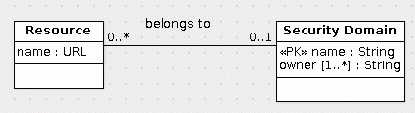
1. (14 points) In an appointment calendar there are periodic events but there can be exceptions (e.g., an event that occurs every month except for December 2010). Design a data model in UML for events. Some events are periodic. Events have descriptions and periodic events have a period. Assume you have a type Duration for a period of time. A periodic event can have exceptions. An exception has a date and may have a reason.



Elements in the UML diagram:

* 1. Event class
  2. description attribute of Event
  3. Periodic Event class
  4. period attribute of Periodic Event
  5. period has type Duration
  6. Exception class
  7. date attribute of Exception class
  8. date has type Date
  9. reason attribute of Exception class
  10. reason attribute has multiplicity [0..1]
  11. Periodic Event is subclass of Event
  12. Association between Periodic Event and Exception
  13. Cardinality cnstraint of Periodic Event end is 1..1
  14. Cardinality cnstraint of Exception end is 0..\*

1. (18 points) Here is a data model written in UML:



Translate this data model to SQL.

create table SecurityDomain (

name varchar(255) primary key

);

create table DomainOwner (

domain varchar(255) references SecurityDomain(name),

owner varchar(255),

primary key(domain, owner)

);

create table Resource (

id int primary key,

name varchar(255) not null,

belongsTo varchar(255) references SecurityDomain(name)

);

alter table SecurityDomain

add foreign key (name) references DomainOwner(domain);

The following are the components of the data model used for grading:

* 1. create table SecurityDomain
  2. name varchar
  3. name is primary key
  4. create table DomainOwner
  5. domain varchar
  6. domain references SecurityDomain(name)
  7. owner varchar
  8. primary key(domain, owner)
  9. create table Resource
  10. id int primary key
  11. name URL or varchar
  12. name is not null and not primary key
  13. belongsTo varchar
  14. belongsTo references SecurityDomain(name)
  15. belongsTo can be null
  16. alter table SecurityDomain
  17. add foreign key (name)
  18. foreign key (name) references DomainOwner(domain)

It is possible to translate the belongs to association to a table. If you used this design, then the explanation for your score might not be precisely correct, but the score is correct nevertheless.

1. Here is an SQL schema:
2. create table Person (
3. id int primary key,
4. name varchar(100)
5. );
6. create table Instrument (
7. name varchar(100) primary key
8. );
9. create table plays (
10. person int references Person(id),
11. instrument varchar(100) references Instrument(name)
12. );
13. create table playsSong (
14. person int,
15. instrument varchar(100),
16. foreign key (person, instrument) references plays(person, instrument),
17. song varchar(200),
18. primary key (person, instrument, song)
19. );

Write the following queries in SQL:

* 1. (10 points) Find the persons by name who play 'Quiet City' on a trumpet.

There are several variations on this query depending on the simplifications one applies. The one with all of the tables is the following:

select p.name

from Person p, Instrument i, plays y, playsSong s

where p.id = y.person

and i.name = y.instrument

and y.person = s.person

and y.instrument = s.instrument

and s.song = 'Quiet City'

and i.name = 'trumpet'

Note that there are 4 join conditions even though there are 4 tables. The reason why there are 4 is that one of the foreign key constraints involves two columns.

If one uses the foreign key constraint of playsSong to eliminate the plays table one has this query:

select p.name

from Person p, Instrument i, playsSong s

where p.id = s.person

and i.name = s.instrument

and s.song = 'Quiet City'

and i.name = 'trumpet'

Finally, one can eliminate the Instrument table to get this query:

select p.name

from Person p, playsSong s

where p.id = s.person

and s.instrument = 'trumpet'

and s.song = 'Quiet City'

* 1. (18 points) Find the instruments that are never used for the song named 'Moldau'.
  2. select i.name
  3. from Instrument i
  4. where not exists (
  5. select \*
  6. from playsSong s
  7. where s.song = 'Moldau'
  8. and i.name = s.instrument

)