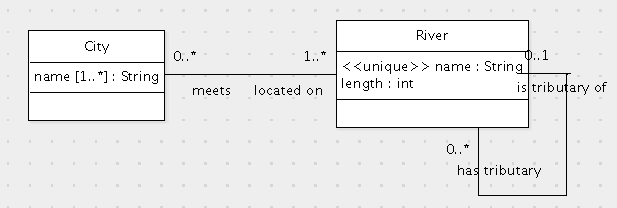
**CS 5200 Database Management Systems Answers to Mid-Term Exam Fall 2010**

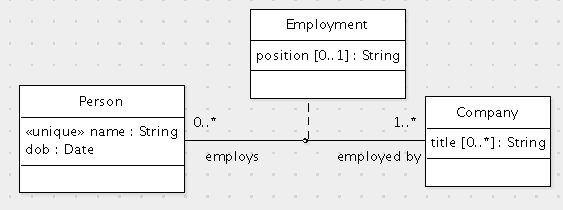
1. (12 points) A geographer has a database of cities that are located on rivers (and only cities like this). A city can be located on several rivers, and a river may meet some cities. Each city has a name, possibly several. A river has a unique name and a length. A river can be the tributary of another river. Design a UML class diagram for the database described above.



Grading Criteria

* 1. City class
  2. name attribute of City class
  3. name attribute of City class has multiplicity 1..\*
  4. River class
  5. name attribute of River class
  6. name attribute of River class is unique
  7. length attribute of River class with appropriate names
  8. length attribute of River class has type double
  9. Association between City and River
  10. Association between City and River has multiplicity 1..\* on River end
  11. Association between City and River has multiplicity 0..\* on City end
  12. Recursive association on River with appropriate names
  13. Recursive association on River has multiplicity 0..1 on one side
  14. Recursive association on River has multiplicity 0..\* on the other side

1. (17 points) Here is a data model written in UML:



Translate this data model to SQL.

create table Person (

id int primary key,

name varchar(255) not null unique,

dob date not null

);

create table Company (

id int primary key

);

create table CompanyTitle (

company int references Company(id),

title varchar(255),

primary key(company, title)

);

create table Employment (

employs int references Person(id),

employedBy int references Company(id),

position varchar(255),

primary key(employs, employedBy)

);

alter table Person

add foreign key(id) references Employment(employs);

Grading criteria:

* 1. create table Person
  2. id int primary key
  3. name is a varchar
  4. name is not null and not primary key
  5. name is unique
  6. dob is a date
  7. dob is not null
  8. create table Company
  9. id int primary key
  10. create table CompanyTitle
  11. company int
  12. company references Company(id) on update cascade on delete cascade
  13. title is a varchar
  14. primary key(company, title)
  15. create table Employment
  16. employs int
  17. employs references Person(id) on update cascade on delete cascade
  18. employedBy int
  19. employedBy references Company(id) on update cascade on delete cascade
  20. position is a varchar
  21. position can be null
  22. primary key(employs, employedBy)
  23. Person.id references Employment(employs)

1. Here is an SQL schema for tracking the time spent by miners in various mines:
2. create table Person (
3. id int primary key,
4. name varchar(100)
5. );
6. create table Mine (
7. designation varchar(100) primary key,
8. location varchar(5000) not null
9. );
10. create table WorkInterval (
11. miner int references Person(id)
12. on update cascade on delete cascade,
13. mine varchar(100) not null references Mine(designation)
14. on update cascade on delete cascade,
15. start datetime not null,
16. end datetime,
17. primary key(miner, start),
18. check (end is null or end > start)
19. );

Write the following queries in SQL:

A. (8 points) For each miner, list the miner's name and the date when the miner first started working in some mine.

select p.name, min(w.start)

from Person p, WorkInterval w

where p.id = w.miner

group by p.id;

Grading criteria:

* 1. select p.name
  2. (2 points) select min(w.start)
  3. from Person p
  4. from WorkInterval w
  5. where p.id = w.miner
  6. (2 points) group by p.id

Some solutions used a nested subquery instead of grouping. This results in duplication in the result set. Unfortunately, specifying distinct will remove required duplicates.

B. (8 points) List the mines by location that have a person named Luis who has worked in the mine.

select m.location

from Mine m

where exists (select \*

from WorkInterval w, Person p

where m.designation = w.mine

and w.miner = p.id

and p.name = 'Luis');

Grading criteria:

* 1. select m.location
  2. from Mine m
  3. where exists (select \*
  4. from WorkInterval w
  5. from Person p
  6. where m.designation = w.mine
  7. and w.miner = p.id
  8. and p.name = 'Luis'

1. (15 points) Using the schema in the previous problem, the following method prints the miners who are currently in a specified mine and how long each has been in the mine (in hours):
2. public void printWorkingMiners(Connection connection, String mine) throws SQLException {
3. long currentTime = System.currentTimeMillis();
4. Date now = new Date(currentTime);
5. PreparedStatement statement = connection.prepareStatement
6. ("select p.name, p.id, w.start from Person p, WorkInterval w " +
7. "where w.miner=p.id and w.mine=? and w.start<? and w.end is null");
8. statement.setString(1, mine);
9. statement.setDate(2, now);
10. ResultSet rs = statement.executeQuery();
11. while (rs.next()) {
12. String name = rs.getString(1);
13. int id = rs.getInt(2);
14. Date start = rs.getDate(3);
15. double durationInHours = (currentTime - start.getTime()) / (1000.0 \* 3600.0);
16. System.out.println(name + " (" + id + ") has been in the mine for " + durationInHours + " hours.");
17. }
18. statement.close();
19. }

Fill in the missing statements.

Grading criteria:

1. (2 points) statement.setString(1, mine);
2. (2 points) statement.setDate(2, now);
3. (3 points) ResultSet rs = statement.executeQuery();
4. (2 points) while (rs.next()) {
5. (2 points) String name = rs.getString(1);
6. (2 points) int id = rs.getInt(2);
7. (2 points) Date start = rs.getDate(3);