Total: 100 marks

1 Question One (50%=50 marks)

Consider the following relational database schema where the underline attributes are the primary keys,

- Lecturer(lId, name, title, address)
- Student(sId, name, major)
- Course(cId, cName, lId)
- Enrolment(cId, sId)

Write relational algebra expressions for the following queries.

1. **(5 marks)** Find the lecturers who teach at least one course. Show the lIds of those lectures.
2. **(6 marks)** Find the students who have the same name as some of lecturers. Show the IDs and names of the students.
3. **(8 marks)** Find the students who do not enrol in any course taught by 'John Smith'. Show the names of those students.
4. **(10 marks)** Find the courses for which there are **not both** students with name 'John' or 'Alice' who enrol in. That is, that in any course appeared in the outcome, there may be a student with name 'John' or a student with name 'Alice', but there should not be both of them. Show the cIds of those courses.
5. **(10 marks)** Find the students who have the same name as some other student. Show the names of the students.
6. **(11 marks)** Find the students who take all the courses taught by 'David Cheung'. Show only the IDs of these students.

2 Question Two (50% = 50 marks)

2.1 Background

Cloud computing is changing the world. It is Internet-based computing: hundreds of thousands of remote computers are interconnected to provide all kinds of resouces, including data, applications, as well as computing power. For example, Google provides online services named *Google Docs*[1], which we may consider as an alternative to Microsoft Office. Google Docs differs from conventional software products in the sense that, when using Google Docs, the data and the software locate no longer in the our own machines but rather somewhere "in the cloud." We need not store the data or install any software. What we have to do is to get a Google Docs account and then simply open an Internet browser. This is also the situation when we rent online movies on Amazon[2]. We pay about 4 dollars to enjoy a movie online. We do not run on the movie in our own machines, nor do we install any movie players. We need only get an Amazon account to do the business.

One of the key observations from the above two examples is that machines become virtual to users in the cloud era. When we are editing online documents or watching online movies, we are also using certain computing power provided by Google or Amazon.

2.2 Problem Description

Now we want to design a database to manage cloud information over the Internet. Let's assume that after a thorough requirements analysis you've got the following understanding about the design goals:

There are several *organizations* (e.g. Google, Amazon, etc.) providing cloud services on the Internet. Each organization has a unique name and an address. Each organization may also maintain its own computers as *virtual machines*, where a virtual machine has a (globally) unique id and a description of its capability (i.e. memory, compute unit, platform, API, etc.). You may store capability as a string. Note that a virtual machine must and can only belong to one organization.

The database also maintains user *accounts*. A user account has a unique account id, as well as some necessary personal information such as email, gender, phone number, date of birth (DOB), etc. For the sake of simplicity, let's assume that all user accounts are universal. That is, let's do not discriminate between, for example, a Google account and an Amazon account – One account deals with everything. However, we do distinguish between two different types of user accounts, the *developer account* and the *customer account*, according to how the account behaves in the system.

A developer account is used to develop *e-products* (e.g. music, movies, etc.). It must and can only be affiliated to one organization, and thus it is always associated with an office address. On the other hand, a customer account can purchase the e-products. Within a customer account the database should maintain the credit card number and the home address of the account owner. The price and discount information of a deal should also be recorded in the database.

An e-product is something that can be sold and consumed over the Internet. Associated to each e-product there is a unique product id as well as the date on which this product was released online. Again for the sake of simplicity, let's consider only four types of e-products: *music*, *book*, *movie*, and *image*. Each type of e-product has their own information. For example, a book should have a title and at least one author. You may figure out something reasonable for each type of e-product by yourself.

Finally, it is important to note that an e-product can only be run (viewed, played, ...) on virtual machines. This also implies that a customer account must have at least one virtual

[1] See https://docs.google.com.
[2] See http://www.amazon.com.

machine. Once a virtual machine is owned by an account, it cannot be owned by other accounts. The date that a virtual machine is owned by some customer should be stored in the database.

2.3 Task

Based on the information above, draw an E-R or E-E-R diagram that models as many concepts and constraints as possible. You should use the notation introduced in the lectures and in the textbook, and provide details of your design choices.

Note

- You may make commonsense assumptions about the situation. For example, an account should have a username. Although it is not mentioned in the description above, it seems a good idea to make username an attribute for accounts. Describe the assumptions you've made, if any.
- It is possible that some of the information above cannot be captured using (E)-E-R model or is irrelevant to database design. Discuss the information that your design cannot capture or you've simply ignored, if any.
- There may be multiple equally good design choices for your (E)-E-R diagram. Provide justification for your design choices and make sure that they are not inconsistent with the database specification.