Assignment 1

Course: COMP378
Name: Jason Bishop
Student#: 3042012

Question 1 (15 marks)
Answer the following questions (250 words max/question).

a. Explain the concept of program-data independence, and explain how it is achieved in the database approach.

Data independence is the concept of separating the description of data from the application programs that use the data. This is achieved in the database approach because all of the data, including its metadata is stored within the database. Data is added, modified and deleted by accessing the database, typically through SQL commands. This allows for the independent storage of data, separate from any programs using the data. Each application program can be developed independently and retrieve the data it needs by accessing the database. Each program can run and do what it is designed to do without having to worry about data maintenance. Each part (program and data) acts independently of one another, and thus, we have program-data independence.

b. Briefly contrast and compare the following development approaches: the systems development life cycle and the prototyping methodology.

The systems development life cycle (SDLC) is a formal method employed by many organizations to develop, maintain, or replace an information system. This usually involves development through the course of a number of steps or phases. Typical phases in the SDLC include planning, analysis, design, implementation and maintenance, though there may be more or less depending on the specific methodology used. Additionally, certain phases may be more closely related than others, and development can move back and forth between phases, but the overall approach is designed to be a top-down, fairly linear process that produces a fully-implemented system ready to meet all user needs. Of course there will be maintenance and issues to iron out, but in theory, this is the idea.

Prototyping on the other hand, works in a different manner. This method is a bottom-up process. The emphasis is on creating a working system that is up and running as soon as possible. This system is continually revised and built upon by close cooperation between the users and the designers/analysts. The system is tested and analyzed at each iteration, then added to with the next iteration. There are no discrete phases or overall plan to follow. However, the end result remains the same: a fully-implemented system ready to meet all user needs.

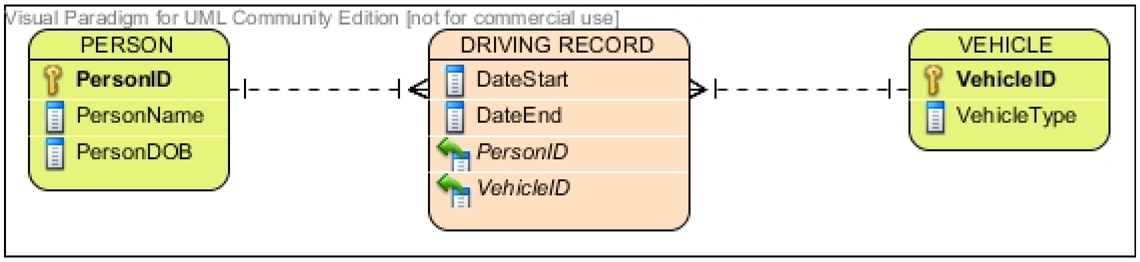
c. Discuss the difference between *entity type* and *entity instance*.

An entity type is defined as a collection of entities that share common properties and characteristics. This is analogous to a category. An example might be a person. Each person has a mother and father, a name, a sex, and an age. Of course, there can be many more characteristics, or there might be only a few, it depends on the entity being defined. An entity instance is defined as a single occurrence of an entity type. There can be many instances of a single type. Each one is an individual unit with unique it's own unique properties. In the bird example above, a specific instance of a person would be me. I am Jason Bishop, my mother is Lee Bishop, my father is Kris Klements. I am a 34 year old male. Another instance might be Scarlett Johansson, a 29 year old female, whose parents are Melanie Sloan and Karsten Johansson (source: http://www.imdb.com/name/nm0424060/bio). For each entity instance, a value must be assigned to each of the properties or characteristics defined in its type, even if that value is 'undefined.'

Question 2 (15 marks)
Give a simple example of an E-R diagram for each of the following concepts:

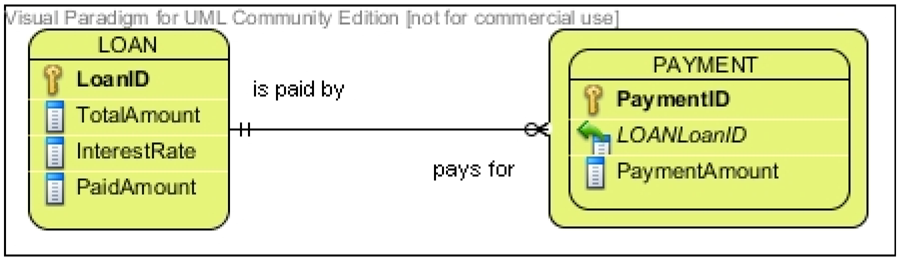
a. Associative entity

An example of an associative entity would be one between a person and a vehicle. Many people can drive a vehicle, and a vehicle can be driven by any number of people. The associative entity would store the information unique to this relationship, namely the time that the car is to be driven. Here is the ERD for this example:



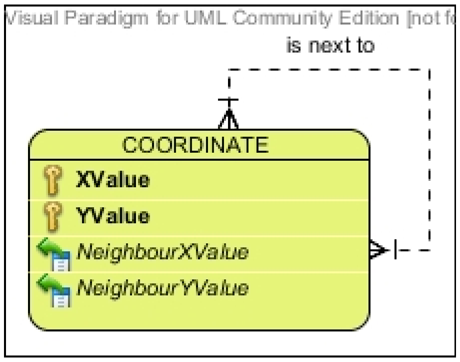
b. Weak entity

An example of a weak entity would be a payment for a loan. The loan would thus be the identifying strong entity for the payment. A person would not be making a payment if they did not have a loan to be paying for. Here is the ERD for this example:



c. Unary relationship

An example of a unary relationship would be the position of a two-dimensional grid co-ordinate, relative to its surrounding co-ordinates. For example, in the positive x-y plane, the origin (0,0), is next to points (0,1) and (1,0). Here is the ERD for this example:



Question 3 (20 marks)
As an independent consultant, you have a contract with Athabasca University to develop an application to support course administration. Read the following detailed description of this application, and complete the tasks listed below.

• A course has a unique course number and title, and is assigned to one or more areas of the Computer Science program. Each course has an instructor, at least one teaching or research assistant, an online conference whiteboard, a time (interval), and a capacity (maximum number of participants). A course need not be offered each term, but the challenge option is open throughout the entire year. A course may have some prerequisite courses, and a student is not allowed to register for a course if s/he has not fulfilled the prerequisites or their equivalent. In addition, a student who would like to challenge a course should have the course prerequisites or authorization from the course instructor.

• An instructor has a first name, last name, email address, phone number, fax number, and is assigned to several areas of expertise. Note that two different instructors might have the same (first and last) names, and they might share the same fax number.

• A student is given a unique student number. For each student, we want to record first name and last name, email address, phone number, postal address, and GPA. Again, the same names may belong to two different individuals. A student may be enrolled in several courses each term. For each enrollment, we want to record the grade. There are two types of student: graduate and undergraduate. Undergraduate students have a study major, while graduate students have both a specialization and a thesis topic.

• A teaching assistant is a graduate student, and we need to record his/her first name, last name, year, and GPA, as well as how many courses (including the current one) the teaching assistant has already taught.

- A research assistant is a student for whom we want to record the usual basic information, plus the number of years of work experience. The research assistant provides laboratory support for students, manages the conference whiteboard, and does some programming tasks.

- An area of study is described by its name. An area may be divided into several sub-areas. These areas are used to categorize instructors, as well as courses. For example, the area of game development consists of sub-areas such as Graphics, Networking, and Human-Computer Interaction; the area of E-Services Technology consists of Artificial Intelligence, Networking, and Distributed Systems.

Design an ER diagram for this course administration system. Draw the complete ER diagrams, including all aspects discussed in the course. Clearly state any further assumptions made, but note that you must not override the specifications above.

Since the final ERD is quite large, I have included it as a separate file. Please see the file named *TME1 Q3.jpg.*
(*Note: As above, this was drawn in Visual Paradigm and represents weak entities as above.)
(**Note: To identify not required attributes, there is an 'N' next to the attribute. Unique attributes are signified by a 'U')

While constructing the ERD I had to make several assumptions. For instance, I assumed that a course could exist without being assigned an instructor, and that an instructor may or may not be assigned to teach a course (ie they could do research). Additionally, I had to assume that an instructor's area of expertise is the same as the area of study that he/she is assigned and which is also assigned to each course. For each course taken by the student, I had to assume that only the final grade needs to be recorded, not individual tests/assignments. Similarly, I assumed that a pass/fail condition would be needed for each challenge attempt by a student. Finally, I had to assume that unique identifiers were needed for each instructor and assistant because these are not mentioned in the specification above, but would probably be necessary to easily identify unique entities.

Question 4 (15 marks)
For each of the following relations, identify the Normal Form(s) each relation satisfies, and transform it into 3NF.

a. Consider the relation STUDENT, where a student can have only one major:

RELATION = STUDENT (StuID, StuName, Major), Primary Key = {StuID}.

In the relation STUDENT, the primary key is identified as StuID and there are no repeating groups in the relation. Thus, the relation is in 1NF. The two non-key attributes, StuName and Major, are both dependent on the primary key. Each StuID identifies a student with one name and one major. Thus, there are no partial functional dependencies, and the relation is in 2NF. Additionally, each of the non-key attributes is not dependent on the primary key via the other non-key attribute. In other words, StuName is not dependent on StuID through Major, and Major is not dependent on StuID through StuName. Thus there are no transitive dependencies, and the relation is in 3NF.

b. Consider the relation EMPLOYEE, where an employee can have more than one specialization:

RELATION = EMPLOYEE (EmpID, Name, Specialization), Primary Key = {EmpID}.

In the relation EMPLOYEE, the primary key is identified as EmpID, however, since there can be multiple instances of Specialization for each EmpID, there can be repeating groups. These repeating groups can be removed by filling in the EmpID and Name information for each repeating Specialization. As it stands, the relation specified is in 1NF. Each non-key attribute (Name, Specialization) is also dependent on the single primary key. Thus, the relation is also in 2NF. Additionally, there are no transitive dependencies in the relation, hence it is also in 3NF.

c. Consider the relation LEASE, where a person can live in only one building, and a building can charge only one rental rate:

RELATION = LEASE (PersonID, BuildingID, Rent), Primary Key – {PersonID}.

In the relation LEASE, the primary key is identified as PersonID and there are no repeating groups in the relation. Thus, the relation is in 1NF. The two non-key attributes, BuildingID and Rent, are both dependent on the single primary key. Thus, it is also in 2NF. However, the attribute Rent is only tranistively dependent on the primary key. It is fully dependent on BuildingID, which is itself dependent on PersonID (PersonID → BuildingID → Rent). If a person moved to a different building charging the same rent, only the BuildingID value would change. Thus, the relation is not in 3NF. To covert it to 3NF, the transitive dependency is removed by creating a new relation, with all relations in 3NF. The relations are as follows:

RELATION = LEASE (PersonID, BuildingID), Primary Key – {PersonID}
RELATION = BUILDING(BuildingID, Rent), Primary Key – {BuildingID}

Each of these relations has a primary key with no repeating groups, hence they are in 1NF. Each relation's non-key attributes are fully dependent on the single primary keys, hence they are in 2NF. Finally, each relation has no transitive dependencies, hence they are in 3NF.

Question 5 (20 marks)
Consider a one-relation database with the following attributes:

Employee number (emp_no), Date hired (date), Job title (job), Phone number (phone_no), Office number (office_no), Area (area), Salary (sal), project number (proj_no), Project budget (p_budget), Department number (dep_no), Department budget (d_budget), and Department manager employee number (mgr_emp_no).

The following business rules apply:

- No employee can manage more than one department at a time.
- No employee can work in more than one department at a time.
- No employee can work on more than one project at a time.
- No employee can have more than one office at a time.
- No employee can have more than one phone at a time.
- No employee can have more than one job at a time.
- No project can be assigned to more than one department at a time.
- No office can be assigned to more than one department at a time.
- Department numbers, employee numbers, project numbers, office numbers, and phone numbers are all "globally" unique.

The following functional dependencies also apply:

- emp_no g phone, emp_no g office_no, emp_no g dep_no, emp_no g proj_no
- {emp_no, date} g job, {emp_no, date} g sal
- phone_no g office_no, office_nog area, office_no g dep_no
- proj_no g dep_no, proj_no g p_budget
- dep_no g mgr_emp_no, dep_no g d_budget
- mgr_emp_no g dep_no

Transform this relation into 3 NF. Justify any decomposition.

Since this is a one-relation table with no repeating groups, it is already in 1NF. The primary key for this relation is {emp_no, date} because it determines all the other attributes directly or transitively. To convert to 2NF, each non-key attribute needs to be fully dependent on the entire key, so the following decomposition is necessary:

Relation 1 = emp_no, phone_no, office_no, proj_no, dep_no, area, p_budget, d_budget, mgr_emp_no
Relation 2 = emp_no, date, job, sal

Here, Relation 2 is in 3NF because there are no transitive dependencies. However, Relation 1 contains several transitive dependencies and can be further decomposed:

Relation 1a = emp_no, phone_no, office, proj_no, dep_no
Relation 1b = office_no, phone_no, area, dep_no
Relation 1c = proj_no, p_budget, dep_no
Relation 1d = dep_no, d_budget, mgr_emp_no

Now, with Relations 1a – 1d and Relation 2, we have all relations in 3NF

Question 6 (15 marks)
Consider the following relations:

Emp(E_id: integer, E_name: string, Age: integer, Salary: real)

Works(E_id: integer, Dep_id: integer, affectation: date)
Dept(Dep_id: integer, Dep_name: string, budget: real, Manager_id: integer)

a. What referential integrity constraints exist between these relations?

The referential integrity constraints in these relations are due to the foreign key values in the Works relation. The foreign keys are E_id and Dep_id. These values correspond the matching names in the Emp and Dept relations respectively. Each of the foreign key attributes must be the primary keys for each of the corresponding relations, and this is indeed the case. Additionally, the values of the primary key attributes must have a value, or be null, but must exist.

b. What are the options for enforcing these constraints when a user attempts to delete a Dept tuple?

If a user tries to delete a Dept tuple, this will have a direct effect on the Works relation. For example, if we have in the Works relation E_id = 1, Dep_id = 1, affectation = 06/06/2006, this will refer to the specific department in the Dept relation with Dep_id = 1. If this department is deleted by a user, the value in the Works relation will not refer to any value in the Dept relation.

We can solve this problem in 3 ways. First, a safety check can be used. This entails making sure all of the referenced values of Dep_id in the Works relation are first deleted before the row in the Dept relation is deleted. Second, a cascading delete can be used. This entails deletion of the chosen row Dept, followed by deletion of all the values in the Works relation that are associated with the same Dep_id. Both of these involve loss of information in the Works relation that may not be wanted. The third option is to place a null value in the Dep_id value in the works relation when the corresponding value is deleted in the Dept relation. All of these solutions do not have any effect on the Emp relation because the referential integrity matters when the E_id is deleted in the Emp relation, not in the Works relation.