Christopher McArthur

40004257

February 6, 2017

Comp353

Assignment 1

Question 1:

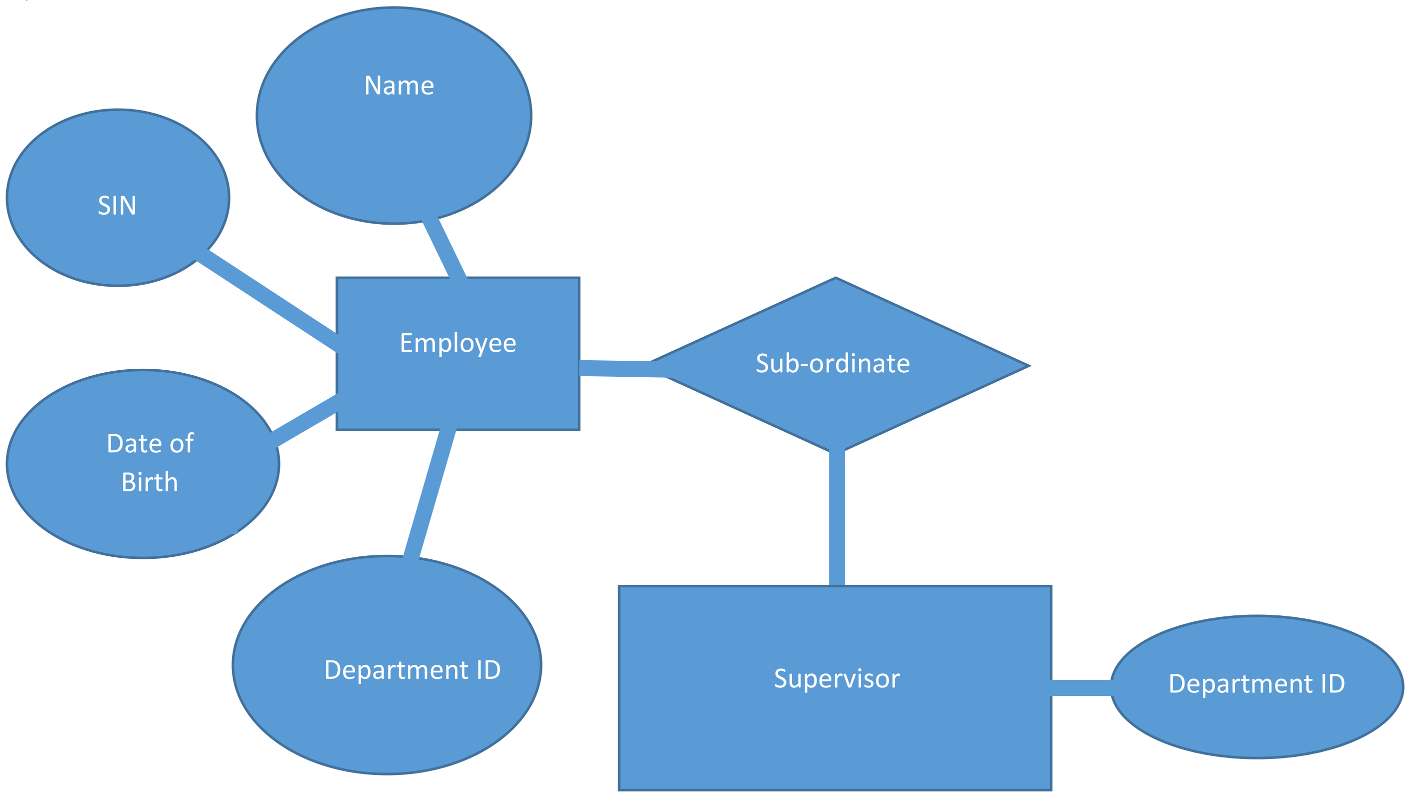
```
#include <stdio.h>

struct Student{
   int SID;
   char[50] Name;
   char[20] Major;
   char[25] email;
}

stuct Course{
   int CID;
   char[50] Name;
   int credits;
}

enum Semester {Winter = 1, Summer, Fall};
enum GradeLetter    {A=1,B,C,D,F};
struct Grade{
   int SID;
   int CID
   int YEAR;
   Semester SEM;
   GradeLetter GRD;

   Grade(int sid, int cid, int year, int sem, int grade)
   : SID(sid), CID(cid), YEAR(year), SEM(sem), GRD(grade)
   { }
}

const char* getLetter( Grade grd)
{
   switch(grd.GRD)
   {
   case 1:
      return "A";
   case 2:
        return "B";
    case 3:
        return "C";
    case 4:
        return "D";
    case 5:
        return "F";
    }
}


void GenerateInfo(FILE* students, FILE* courses, FILE* grades)
{
    //-------------------------------------------------------------------------------------
    students = fopen ("students.txt" , "ra");
    if (pFile == NULL) perror ("Error opening file");
    else
    {
        const char* name = "John_Smith";
        const char* major = "Comp_Sci";
        const char* email = "johnsmith@example.com
        for(int i = 1; i <= 50; i+=1)
        {
            fprintf (students, "%d,%s,%s,%s\n" , i , name , major , email);
        }
    }
    fclose(students);
    //-------------------------------------------------------------------------------------
    courses = fopen ("courses.txt" , "ra");
    if (pFile == NULL) perror ("Error opening file");
    else
    {
        const char* course = "course name";
        const int credits = 3;
        for(int i = 1; i <= 10; i+=1)
        {
            fprintf (courses, "%d,%s,%d\n" , i , course , credits);
        }
    }
    fclose(courses);
    //-------------------------------------------------------------------------------------
    grades = fopen ("grades.txt" , "ra");
    if (pFile == NULL) perror ("Error opening file");
    else
    {
        // I set this to a thousand because in testing at 100 I got 0 results
        for(int i = 1; i <= 1000; i+=1)
        {
            int SID = rand()%50+1;
            int CID = rand()%10+1;
            int year = rand()%25+1990;
            int SEM = rand()%3+1;
            int GRD = rand()%5+1;

            fprintf (grades, "%d,%d,%d,%d,%d\n" , SID , CID , year, SEM, GRD);
        }
    }
    fclose(grades);
    //-----------------------------------------------------------------------------------------
}

////////////////////////////////////////////////////////////////////////////////////////////
//
// Seach for students who have taken class 5 and select name and grade with grade A or B
//
////////////////////////////////////////////////////////////////////////////////////////////

int main()
{
   FILE* students;
   FILE* course;
   FILE* grades;

   void GenerateInfo(students, courses, grades);

    //-----------------------------------------------------------------------------------------

    Student allStudents[50];
    students = fopen ("students.txt" , "r");
    if (students == NULL) perror ("Error opening file");
    else
    {
        int output = 0;
        char strA[50] = NULL;
        char strB[20] = NULL;
        char strC[25] = NULL;
        for(int 1 = 0; i < 50; i += 1)
        {
            fscanf (students, "%d", &output);
            fscanf (students, "%s", strA);
            fscanf (students, "%s", strB);
            fscanf (students, "%s", strC);

            allStudents[i] = Students(outputs, strA, strB, strC);
        }
        fclose(students);
    }

    //-------------------------------------------------------------------------------------------

    Grade allGrades[1000];
    grades = fopen ("grades.txt" , "r");
    if (grades == NULL) perror ("Error opening file");
    else
    {
        int SID;
        int CID;
        int year;
        int SEM;
        int GRD;

         for(int 1 = 0; i < 1000; i += 1)
         {
            fscanf (grades, "%d", &SID);
            fscanf (grades, "%d", &CID);
            fscanf (grades, "%d", &year);
            fscanf (grades, "%d", &SEM);
            fscanf (grades, "%d", &GRD);

            allGrades[i] = Grade(SID, CID, year, SEM, GRD);
         }
         fclose (grades);
    }

    printf("Searching for matches...\n")
    for(int i = 0; i < 1000; i+=1)
    {
        if(allGrade[i].CID == 5 && allGrade[i].GRD <= 2)
        {
            for(int j = 0; j < 50; j += 1)
            {
                if( allStudents[j].SID == allGrade[i].SID )
                {
                    printf("student: %s recieved grade: %s\n", allStudents[j].Name, getLetter( allGrade[i]) );
                }
            }
        }
   }
   printf("\n\nend of program");
   return 0;
}
```

This is very very clearly an ridiculous amount of code for nothing and its SLOW … using SQL is a god sent

SELECT student.name, Grade.letter FROM student, Grade where (grade.CID == 5) && (grade.letter == A or grade.letter B) && (grade.SID == student.SID)

The only advantage for written code is that you can trace it easier, however with MySQL's C++ connector is super easy to use! Plus, it allows you to have both traceable code with a DBMS in the background.

Question 2:



This module assumes all supervisors are employees

Employee			
Name	SIN	Date of Birth	Department ID

Supervisor	
Employee	Department ID

Question 3:

Product	
Model Number(#)	EquipementType(PC/Laptop/Printer)

PC				
Model Number(#)	Processor (GHz)	RAM(Mb)	HDD(Gb)	Price($)

Laptop					
Model Number(#)	Processor (GHz)	RAM(Mb)	HDD(Gb)	Screen(in)	Price($)

Printer			
Model Number(#)	Color(bool)	Type(Laser/Ink)	Price($)

Only referencing by Model Number is risky since it's very unlikely duplicates will not a raise with many manufactures are at play. Model Number and Brand is good solution but including version number may be another great help.

1)
```
CREATE TABLE PC (
        Model int PRIMARY KEY,
        CPU float not null,
        RAM int not null,
        HDD int,
        Price float,
        PRIMARY KEY (Model)
        )
CREATE TABLE Laptop(
        Model int PRIMARY KEY,
        CPU float not null,
        RAM int not null,
        HDD int,
        Screen int,
        Price float
        PRIMARY KEY (Model)
        )
CREATE TABLE Printer(
        Model int PRIMARY KEY,
        Color bool,
        Type enum { 'Laser' , 'Ink' }
        Price float
        PRIMARY KEY (Model)
        )
CREATE TABLE Product(
        Model int PRIMARY KEY,
        Type enum { 'PC' , 'Laptop' , 'Printer' }
        PRIMARY KEY (Model),
        FOREIGN KEY (Model) REFERENCES PC(Model),
        FOREIGN KEY (Model) REFERENCES Laptop(Model)
        FOREIGN KEY (Model) REFERENCES Printer(Model)
        )
```

I don’t know if this will be accepted as it’s a *Polymorphic Associations.* Theses are usually unsupported in SQL due to their nature.

2)

```
ALTER TABLE Printer DROP COLUMN Color;
```

3)

ALTER TABLE Laptop ADD COLUMN odt emun { ‘none’ , ‘cd’ , ‘dvd’ } AFTER hdd;

Here extra because I made a mistake =D

PC				
Model Number(#)	Processor (GHz)	RAM(Mb)	HDD(Gb)	Price($)
589456984	2.8	56454	546	654.56
355455423	4.9	68545	5867	654984.88
165546835	3.5	58498	899	5984.98
684798461	6.5	38898	54948	8494.84
879846545	3.3	65984	6549	6854.35

Laptop					
Model Number(#)	Processor (GHz)	RAM(Mb)	HDD(Gb)	Screen(in)	Price($)
5484987984	1.2	6854	498	28.8	894984.55
4984989494	8.7	684984	5849845	55	98498498498484
6849494989	6.5	69849	849849	23.1	65489.65

Printer			
Model Number(#)	Color(bool)	Type(Laser/Ink)	Price($)
2335452342342	True	Laser	6516948.65
3452345324534	False	Ink	5649.55
3452366856743	False	Laser	564688.48
2454578428536	True	Laser	5698.26
6356826826887	False	Ink	321982.65
6583658387362	true	ink	7899813.59

Question 4:

Account = {AccNumber, AccType, Balance, OpenDate}
Customer = {CustID, FirstName, LastName, Address, AccNumber}

1)
AccNumber(100000-999999)
AccType(‘saving’ , ‘checking’)
Balance(-999999 to 999999)
OpenDate(“yyyy-mm-dd”)

CustID(100000-9999999)
FirstName(varchar[50])
LastName(varchar[50])
Address(varchar[50])
AccNumber(100000-999999)

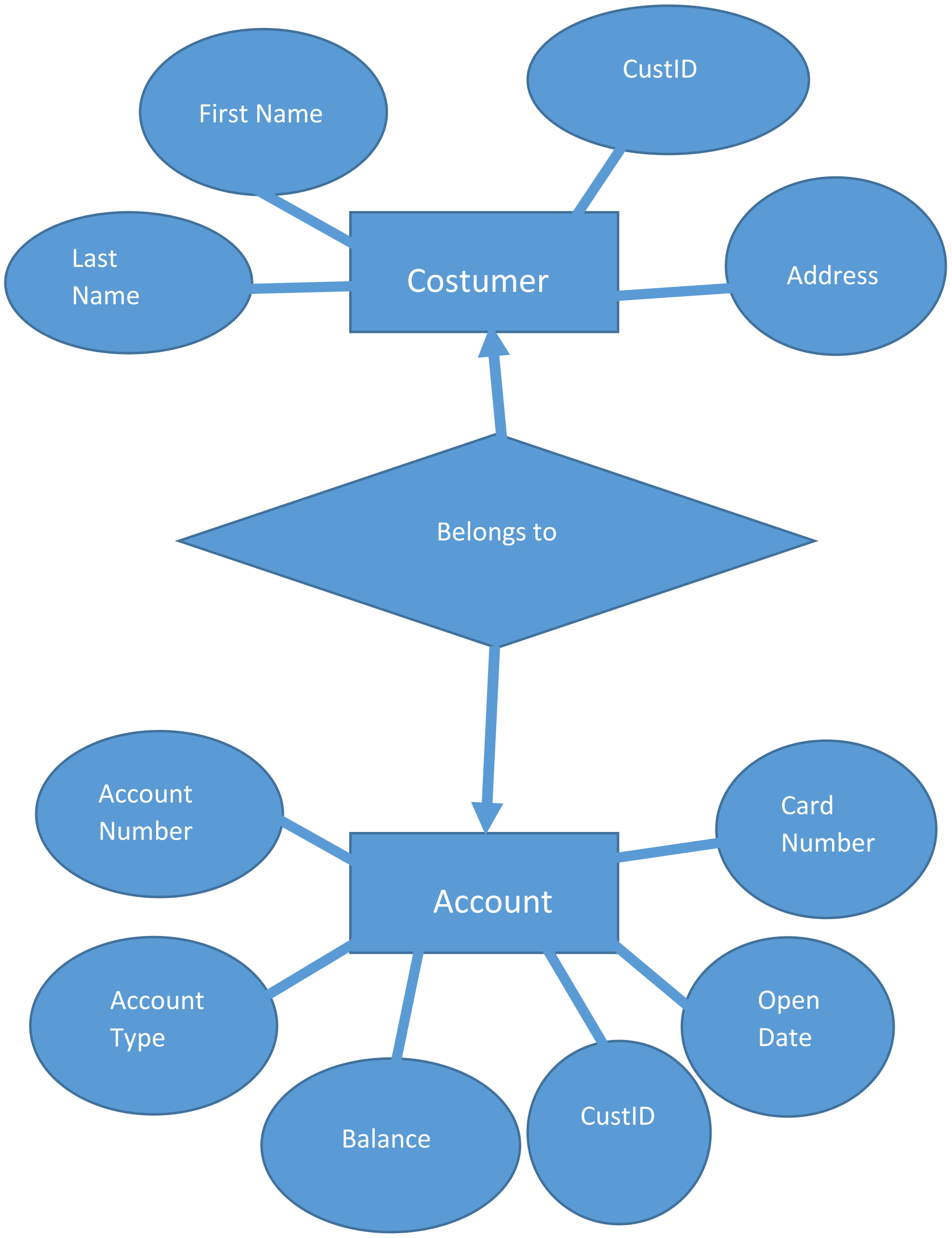
2)

Account			
AccNumber	AccType	Balance	OpenDate
165468	Saving	168498	1988-05-18
984916	Checking	21354	2005-11-31
847498	Checking	6549	1997-10-29
289139	Saving	-15646	2017-03-05

Costumer				
CustID	FirstName	LastName	Address	AccNumber
16584	John	Smith	1442 Main st.	165468
56484	Joe	Smith	1442 Main st.	984916
78165	Jane	Smith	1442 Main st.	847498
32589	Josh	Smith	1442 Main st.	289139

3)

A customer is mostly likely have more than one account ➔ accounts should have a CustID, not Costumer having AccNumber. Even that more than one costumer can have access to an account (I don’t feel like fixing that). There are missing account types, like mortgage, TFSA, and credit. Also debit/credit cards are attached to accounts.


First Name
CustID
Last Name
Costumer
Address
Belongs to
Account Number
Card Number
Account
Account Type
Open Date
Balance
CustID