# CmpE 321
# Introduction to Database Systems
Spring 2014
Project 2

Student Name: Evin Pinar Ornek
Student ID: 2012400057

April 29, 2015

# Contents

**1 Introduction** **3**

**2 Algorithms** **3**

2.1 The Main Algorithm . . . . . 3

2.2 Data Definition Language . . . . . 3

2.2.1 Create a Type of File . . . . . 4

2.2.2 Delete a File . . . . . 4

2.2.3 Display a File . . . . . 4

2.2.4 Display All Files . . . . . 5

2.3 Data Manipulation Language . . . . . 5

2.3.1 Insert a Record . . . . . 5

2.3.2 Retrieve a Record . . . . . 6

2.3.3 Retrieve All Record . . . . . 6

2.3.4 Delete A Record . . . . . 7

2.3.5 Delete All Records . . . . . 7

**3 Conclusion** **7**

**4 Appendix** **8**

# 1 Introduction

In the first project, there was a description for a simple storage management system design. In this project, there is an implementation of the design in C++ programming language. The overall system runs a menu-driven application. The program obtains input from a user by displaying the menu. The menu consists of options, from which the user indicates her choice. To use the application, the program, the data files and the system catalogue -which all have explained in the previous project- must be in the same directory. However, the application does not require to have the information about the files. It just does the request of the user. Generally, there are two types of requests, DDL and DML. The DDL option lets user to create, delete and display a file. The latter one allows user to insert, retrieve and delete a record. There are not any error checking process. The READ-ME document is attached with the project CD-ROM.

# 2 Algorithms

## 2.1 The Main Algorithm

```
/* Asks user to choose one of the options.                    */
while TRUE do
    show the menu of options;
    ask user to choose DDL or DML;
    if DDL then
        go to DDL function;
    else
        go to DML function;
    end
end
```

## 2.2 Data Definition Language

A data definition(or description) language(DDL) is a syntax to define data structures.

### 2.2.1 Create a Type of File

```
/* Creates a type of file.The details are asked from user.The initial
   file is null.                                                    */
```
open system catalog file;
find the next empty record space;
ask file name or type;
create a data file with that name;
ask # of fields;
**for** $i = 1$ *to # of fields* **do**
    ask the field name;
    ask the field size;
**end**
ask the primary key value;
update the system catalog file;
set the # of empty records to maximum;

### 2.2.2 Delete a File

```
/* Deletes a type of file by its name.                              */
```
open system catalog file;
ask the file name to be deleted;
iterate in the system catalog;
**if** *recordName* == *fileName* **then**
    remove the record from the system catalog file;
    delete the record;
**end**
update the system catalog file;
close the system catalog file;

### 2.2.3 Display a File

```
/* Displays a type of file and its format.                          */
```
open system catalog file;
ask the file name to be displayed;
iterate in the system catalog;
**if** *recordName* == *fileName* **then**
    **print** the file name;
    **print** number of fields;
    **print** field names and sizes;
    **print** the primary key;
**end**
close the system catalog file;

### 2.2.4 Display All Files

```
/* Displays all types of files and their formats.                    */
```
open system catalog file;
iterate in the system catalog;
**foreach** *type of file in the system catalog* **do**
    **print** the file name;
    **print** number of fields;
    **print** field names and sizes;
    **print** the primary key;
**end**
close the system catalog file;

## 2.3 Data Manipulation Language

A data manipulation language(DML) is used for selecting, inserting, deleting and updating data in a database.

### 2.3.1 Insert a Record

```
/* Creates a type of record.The field informations are asked from user.
   */
```
open system catalog file;
ask file name or type to insert the record;
open data file with that name;
**foreach** *page in the data file* **do**
    **if** *the page has empty record space* **then**
        get the next empty record to memory;
    **else**
        create another page;
        get the next empty record to memory;
    **end**
**end**
**for** $i = 1$ *to* # *of fields* **do**
    ask the field information;
**end**
ask the key value for the new record;
update the record;
update the page header;
close the data file;
close the system catalog file;

### 2.3.2 Retrieve a Record

```
/* Retrieves and displays a record with a given primary key value and
   file type.                                                        */
open system catalog file;
ask file name or type of the record;
open data file with that name;
ask the primary key value of the record;
foreach page in the data file do
    if nextPrimaryKey == myPrimaryKey then
        for i = 1 to # of fields do
            print field information;
            nextLine;
        end
    end
end
close the data file;
close the system catalog file;
```

### 2.3.3 Retrieve All Record

```
/* Retrieves and displays all record in a given file type.          */
open system catalog file;
ask file name or type of the record;
open data file with that name;
foreach page in the data file do
    foreach record in the page do
        print primary key value;
        for i = 1 to # of fields do
            print field information;
        end
        nextLine;
    end
end
close the data file;
close the system catalog file;
```

### 2.3.4 Delete A Record

```
/* Deletes a record with a given primary key value and file type.     */
```
open system catalog file;
ask file name or type of the record;
open data file with that name;
ask the primary key value of the record;
**foreach** *page in the data file* **do**
    **if** *nextPrimaryKey* == *myPrimaryKey* **then**
        delete the record;
    **end**
**end**
update the page header; close the data file;
close the system catalog file;

### 2.3.5 Delete All Records

```
/* Deletes all records with in the given file type.                   */
```
open system catalog file;
ask file name or type of the record;
open data file with that name;
**foreach** *page in the data file* **do**
    **foreach** *record in the page* **do**
        delete the records;
    **end**
**end**
update the page header; close the data file;
close the system catalog file;

# 3 Conclusion

This project applies the design of the simple storage management system. The design offers a basic approach to solve the problem of a storage. Therefore, the application is also very simple and primitive. It allows limited number of operations to user and there are many restrictions. There are not any GUI. Due to the prohibition of error checking, the program presumes that the user knows how to use the program perfectly. In a case of error, the system just disrupts. To prevent the errors made by user, I have tried to write down the explanations in a clear way. Also I have provided a READ-ME document with the CD.

# 4 Appendix

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdio>
#include <sstream>

using namespace std;

string syscat = "syscat.dat";

void createFile(){
        ofstream out;
        out.open(syscat, ios::app);
        string fileName, fieldName; int fieldnum, fieldSize, key;
        cout<<"What is the file name?(Please write it without blank) "; cin
        cout<<"What is the number of fields?(Max 10) "; cin>>fieldnum; out<
        cout<<"Which one is the primary key? "; cin>>key; out<<key<<" ";
        cout<<"What is the field size? (Max 29 bytes) "; cin>>fieldSize; ou
        cout<<"What are the field names? "<<endl;
        for(int i=0; i<fieldnum; i++){
                cout<<(i+1)<<". Field: ";
                cin>>fieldName;
                out<<fieldName<<" ";
        }
        out<<endl;
        out.close();

        ofstream out2;
        out2.open(fileName);
        out2<<30; //Number of empty records.
        out2.close();
}

void deleteFile(){
        string fileName, line, name, temp;
        temp="temp";
        cout<<"What is the file name you want to delete? "; cin>>fileName;
        ifstream in(syscat);
        ofstream out(temp);
```

```
        while(getline(in, line)){
                istringstream iss(line);
                iss>>name;
                if(name!=fileName){
                        out<<line<<"\n";
                }
        }
        in.close();
        out.close();
        remove(syscat.c_str());
        rename(temp.c_str(), syscat.c_str());
        remove(fileName.c_str());
}

void displayFile(){
        string fileName, fieldName, line, name, fieldNum, fieldSize, key; i
        cout<<"What is the file name? "; cin>>fileName;
        ifstream in(syscat);
        while(getline(in, line)){
                istringstream iss(line);
                iss>>name;
                if(name==fileName){
                        iss>>fieldNum;
                        fieldNumb=atoi(fieldNum.c_str());
                        iss>>fieldSize;
                        cout<<"Number of fields: "<<fieldNum<<" -> ";
                        for(int i=1; i<fieldNumb; i++){
                                iss>>fieldName;
                                cout<<fieldName<<" - ";
                        }
                        iss>>fieldName;
                        cout<<fieldName<<endl;
                        cout<<"Size of fields: "<<fieldSize<<" bytes"<<endl
                        iss>>key;
                        cout<<"The primary key valu : "<<key<<endl;
                }
        }
        in.close();
}

void displayAllFiles(){
        int counter=1;
```

```
        string fileName, fieldName, line, name, fieldNum, fieldSize, key; i
        ifstream in(syscat);
        while(getline(in, line)){
                cout<<endl;
                istringstream iss(line);
                iss>>name;
                cout<<counter<<". File name: "<<name<<endl;
                iss>>fieldNum;
                fieldNumb=atoi(fieldNum.c_str());
                iss>>key;
                cout<<"The primary key value : "<<key<<endl;
                iss>>fieldSize;
                cout<<"Number of fields: "<<fieldNum<<" -> ";
                for(int i=1; i<fieldNumb; i++){
                        iss>>fieldName;
                        cout<<fieldName<<" - ";
                }
                iss>>fieldName;
                cout<<fieldName<<endl;
                cout<<"Size of fields: "<<fieldSize<<" bytes"<<endl;
                counter++;
        }
        in.close();
}

void insertRecord(){
        string fileName, line, name, fields, record, temp; int emptyRecord=
        cout<<"The files are listed above, please choose a type to insert y
        displayAllFiles();
        cout<<"What is the file name to insert a new record?: "; cin>>fileN
        ifstream in1(syscat); // Check the number of fields from the syscat
        while(getline(in1, line)){
                istringstream iss(line);
                iss>>name;
                if(name==fileName){
                        iss>>fieldNum;
                }
        }
        in1.close();

        record=""; //Take the input.
        for(int i=1; i<=fieldNum; i++){
```

```
            cout<<"Please type the "<<i<<". field: ";
            cin>>fields;
            record=record + " " + fields;
    }

    temp="temp";
    ifstream in(fileName);
    ofstream out(temp);

    while(getline(in, line)){
            istringstream iss(line);
            iss>>name;
            isEmpty=1;
            emptyRecord=atoi(name.c_str());
            if(emptyRecord==0){ // The maximum records in a page is 30
                    out<<emptyRecord<<"\n";
                    for(int i=0; i<30; i++){
                            getline(in,line);
                            out<<line<<"\n";
                    }
                    isEmpty=0;
            }
            else{ // If the page is empty, we decrease "empty page" fro
                    emptyRecord--;
                    out<<emptyRecord<<"\n";
                    for(int i=1; i<=30-(emptyRecord+1); i++){
                            getline(in, line);
                            out<<line<<"\n";
                    }
                    isEmpty=1;
            }
    }
    if(isEmpty==0){
            out<<29<<"\n";
            out<<record<<"\n";
    }else{
            out<<record<<"\n";
    }

    in.close();
    out.close();
    remove(fileName.c_str());
```

```
        rename(temp.c_str(), fileName.c_str());
}

void retrieveRecord(){
        string fileName, line, name, field, temp, keyValue;
        int fieldNum, primaryKey, emptyRecord;
        bool isOk, findIt=0;//
        cout<<"\n All of the files are listed below. \n";
        displayAllFiles();
        cout<<"What is the file name of the record? "; cin>>fileName;
        ifstream in(syscat);  //Find the primary key from the system catalo
        while(getline(in, line)){
                istringstream iss(line);
                iss>>name;
                if(name==fileName){
                        iss>>fieldNum;
                        iss>>primaryKey;
                        break;
                }
        }
        in.close();
        cout<<"What is the primary key value of the record? ";
        cin>>keyValue;
        ifstream in1(fileName); //Find the record with a given primary key
        while(getline(in1, line) && findIt==0){
                istringstream iss(line);
                iss>>name;
                emptyRecord=atoi(name.c_str());
                for(int i=1; i<=30-emptyRecord; i++){  // Find the field w
                        getline(in1, line);
                        istringstream iss1(line);
                        for(int i=1; i<=primaryKey; i++){
                                iss1>>field;
                        }
                        if(field==keyValue){
                                istringstream iss2(line);
                                for(int i=1; i<=fieldNum; i++){
                                        iss2>>field;
                                        cout<<i<<". Field : "<<field<<endl;
                                }
                                findIt=1;
                        }
```

```
                }
        }
        in1.close();
}

void retrieveAllRecords(){
        string fileName, line, name, field;
        int fieldNum, curr=0, emptyRecord;
        bool isOk;
        cout<<"\n All of the files are listed below. \n";
        displayAllFiles();
        cout<<"What is the file name of the record? "; cin>>fileName;
        ifstream in(syscat);  //Find the primary key from the system catalo
        while(getline(in, line)){
                istringstream iss(line);
                iss>>name;
                if(name==fileName){
                        iss>>fieldNum;
                        break;
                }
        }
        in.close();
        ifstream in1(fileName);
        while(getline(in1, line)){ //Output all the records
                istringstream iss(line);
                iss>>name;
                emptyRecord=atoi(name.c_str());
                for(int i=1; i<=30-emptyRecord; i++){
                        getline(in1, line);
                        curr++;
                        cout<<endl;
                        cout<<curr<<". Record's fields: \n";
                        istringstream iss1(line);
                        for(int j=1; j<=fieldNum; j++){
                                iss1>>field;
                                cout<<j<<". Field : "<<field<<endl;
                        }

                }
        }
        in1.close();
}
```

```
void deleteRecord(){
        string fileName, line, name, field, temp, keyValue, temp1;
        int fieldNum, primaryKey, emptyRecord, lineCnt=0, emptyRecLine;//L
        bool isOk, findIt=0;//
        temp="temp";
        temp1="temp1";
        cout<<"\n All of the files are listed below. \n";
        displayAllFiles();
        cout<<"What is the file name of the record to delete? "; cin>>fileN
        ifstream in(syscat);  //Find the primary key from the system catalo
        ofstream out(temp);
        while(getline(in, line)){
                istringstream iss(line);
                iss>>name;
                if(name==fileName){
                        iss>>fieldNum;
                        iss>>primaryKey;
                        break;
                }
        }
        in.close();
        cout<<"What is the primary key value of the record to delete? ";
        cin>>keyValue;
        ifstream in1(fileName); //Find the record with a given primary key
        while(getline(in1, line)){
                istringstream iss(line);
                iss>>name;
                emptyRecord=atoi(name.c_str());
                out<<emptyRecord<<"\n";
                for(int i=1; i<=30-emptyRecord; i++){  // Find the field w
                        getline(in1, line);
                        istringstream iss1(line);
                        findIt=0;
                        for(int i=1; i<=primaryKey; i++){
                                iss1>>field;
                        }
                        if(field==keyValue){
                                istringstream iss2(line);
                                for(int i=1; i<=fieldNum; i++){
                                        iss2>>field;
                                        cout<<i<<". Field : "<<field<<endl;
```

```
                                        }
                                        findIt=1;
                                        emptyRecLine=lineCnt;
                                }
                                lineCnt++;
                                if(!findIt){
                                        out<<line<<"\n";
                                }
                        }
                }
                in1.close();
                out.close();
                //Now, change the header, the "empty record" value.
                lineCnt=0;
                emptyRecLine/=31;
                ifstream in2(temp);
                ofstream out1(temp1);
                while(getline(in2,line)){
                        istringstream iss2(line);
                        iss2>>name;
                        emptyRecord=atoi(name.c_str());
                        if(lineCnt==emptyRecLine){
                                out1<<(emptyRecord+1)<<"\n";
                        }else{
                                out1<<emptyRecord<<"\n";
                        }
                        for(int i=1; i<=30-emptyRecord; i++){
                                getline(in2,line);
                                out1<<line<<"\n";
                        }
                        lineCnt++;
                }
                in2.close();
                out1.close();
                remove(fileName.c_str());
                rename(temp1.c_str(), fileName.c_str());
                remove(temp.c_str());
}

void deleteAllRecords(){
        string fileName, line, name, temp;
        temp="temp";
```

```
        cout<<"\n All of the files are listed below. \n";
        displayAllFiles();
        cout<<"Which records of a file you would like to delete? ";
        cin>>fileName;
        ofstream out(temp);
        out<<30; //Number of empty records.
        remove(fileName.c_str());
        rename(temp.c_str(), fileName.c_str());
}

void goToOperation(int opt){
        switch(opt){
                case 1 : createFile(); break;
                case 2 : deleteFile(); break;
                case 3 : displayFile(); break;
                case 4 : displayAllFiles(); break;
                case 5 : insertRecord(); break;
                case 6 : retrieveRecord(); break;
                case 7 : retrieveAllRecords(); break;
                case 8 : deleteRecord(); break;
                case 9 : deleteAllRecords(); break;
        }
}

int main(){
        int opt=0;
        bool cont=1;
        cout<<"\n\n\nWelcome to the Simple Storage Management System ! \n\n
        while(cont){
                cout<< "There are some operations for Data Definition Langu
                cout<< "DDL options:\n\t1. Create a Type of File\n\t2. Del
                cout<< "DML options:\n\t5. Insert a Record\n\t6. Retrieve a
                cout<<"\n\nNumber of your option : ";
                cin>>opt;
                goToOperation(opt);
                cout<<"\nWould you like to continue or exit? (0/1) ";
                cin>>cont;
        }
        cout<<"\n\nGoodbye !\n\n"<<endl;
        return 0;
}
```