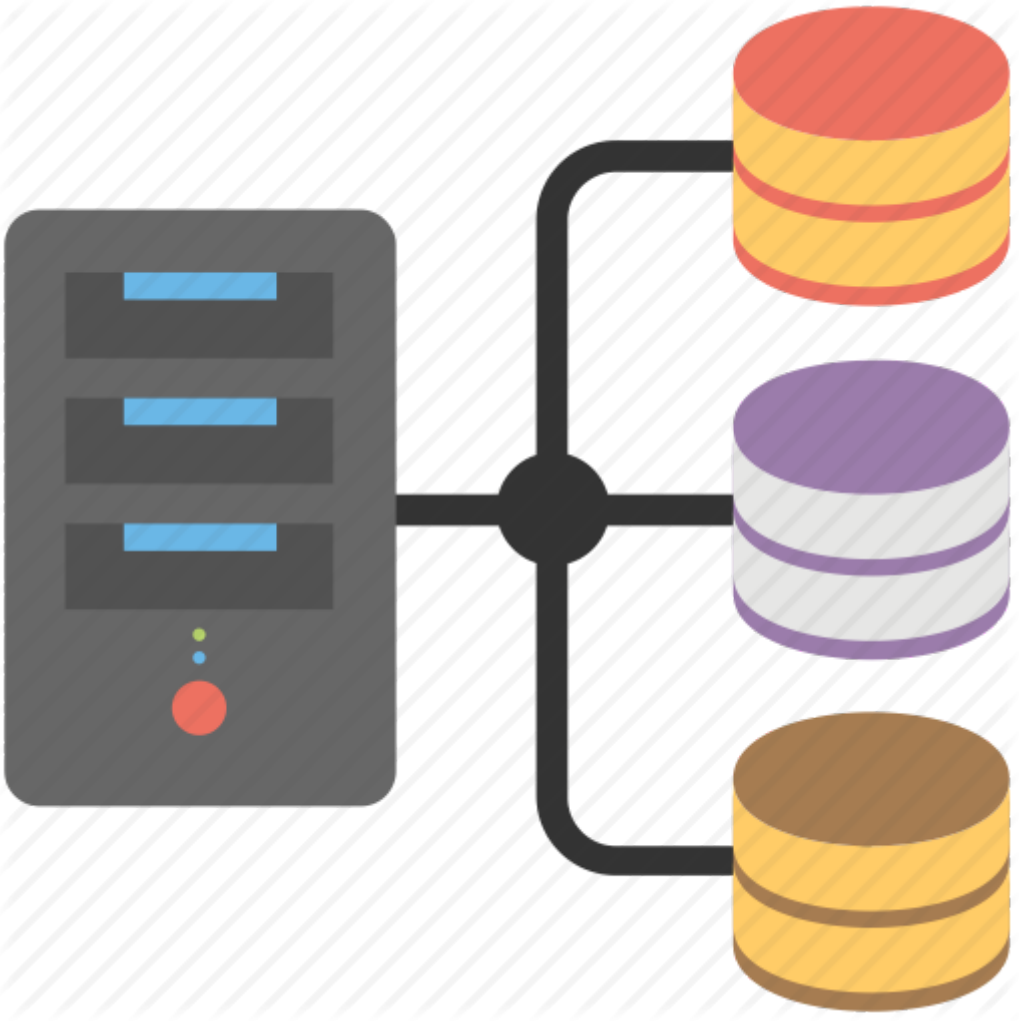
CmpE 321 - 2020/Summer

Project - 2:

Simple Storage Management System



Prepared by:

Hazar ÇAKIR - 2017400093

I) Introduction:

This is a design project that is the second part of the greater project. We are expected to code a storage management system basically. System should fulfill some basic type operations such as creating a type, deleting a type and list all types and some record operations such as creating a record, deleting a record, searching a record and listing all records of a type. While coding such a system, there will be some assumptions and constraints which will be described in detail in the next chapter.

Same assumptions are held but I have some changes Which will be indicated.

II) Assumptions & Constraints:

- In the System Catalog, pointer to the file field has vanished because every files name is as same as Type Name.
- In the page header, pointer to the next page field stores a 1 byte number which constraints the max number of page to 256.

III) Storage Structures:

a) System Catalogue:

System Catalogue									
1 byte	**20 bytes**	**1 byte**	**20 bytes each field name (32 * 20 = 640 bytes)**						**1 byte**
Deleted	**Type Name**	**Number of Fields**	**Name_of_Fields**						**Size_of a_Record**
0-1	Type name 1	n	Field 1	Field 2	Field 3	...	Field n-1	Field n	4*n + 1 bytes
0-1	Type name 2	15	Field' 1	Field' 2	Field' 3	...	Field' 15	Empty...	61 bytes
0-1	Type name 3	2	Field'' 1	Field'' 2	Empty	...	Empty	Empty	9 bytes
0-1	Type name 4	6	Field''' 1	Field''' 2	Field''' 3	...	Empty	Empty	25 bytes

...	...	...	...	...	...	...	...	...	..

```
def createType():
   file = open("Sys_cat.txt","r+",encoding="latin-1")
   print("Name of the type: ( min 8 character -- max 20 character ): ")
   while True:
       typeName = input()
       if len(typeName) < 8 or len(typeName) >20:
            print("Invalid name, please type min 8 character -- max 20
character")
       else:
           break
   control = findType(typeName)
   if isinstance(control, list) :
        print("There is a type already with same name,Type Name should
be unique")
       return
   print("Number of Fields ( min 3 -- max 32):")
   while True:
       numOfFields = int(input())
       if numOfFields < 3 or numOfFields > 32:
           print("Please type a valid number: ( min 3 -- max 32)")
       else:
           break
   namesOfFields = []
   print("Please type field names one by one: ")
   for i in range(numOfFields):
       print("Field " + str(i+1) +" : ")
       while True:
           fieldName = input()
           if len(fieldName) < 8 or len(fieldName) >20:
                print("Invalid name, please type min 8 character -- max
20 character")
           else:
               namesOfFields.append(fieldName)
               break

   newType = makeType(typeName,numOfFields, namesOfFields)
```

```
    page = 1
    content = diskDrive("Sys_cat",page)
    while True:
        pos = 0
        while pos + 663 < 2048:
            if content[pos] != chr(0):
                pos += 663
            else:
                content[pos:pos+663] = list(newType)
                file.seek((page-1)*2048)
                file.write("".join(content))
                file.close()
                createFile(newType)
                print("Type created successfuly")
                return
        page += 1
        content = diskDrive("Sys_cat",page)

def makeType(typeName,numOfFields,namesOfFields):
    myType = "0"

    for i in range(20 - len(typeName)):
        typeName += " "
    myType += typeName + chr(numOfFields)

    for i in range(32):
        if i < numOfFields:
            myType += namesOfFields[i]
            for k in range(20-len(namesOfFields[i])):
                myType += " "

        else:
            for k in range(20):
                myType += " "
    myType += chr(4*numOfFields+1)

    return myType
```

b) Page Header and Page Structure:

Page Header				Rest of the Page		
1 byte	1 byte	2 byte	2 byte			
Full	Pointer_to the_Next Page	Number_of Records	Max_Number_of Records			
1-0	0-255	15	200	Record 1	Record 2	...

```
def createFile(typeInfo):
    content = list(typeInfo)
    name = "".join(content[1:21]).strip()
    file = open( name+ ".txt","w",encoding="latin-1")
    file.write("0")
    file.write(chr(2))
    file.write(chr(0))
    file.write(chr(0))
    number = math.floor((2042/ord(typeInfo[-1])))
    file.write(intToChar(number,2))
    for i in range(2042):
        file.write(chr(0))
    file.close()


def createNewPage(typeInfo,page):

    content = list(typeInfo)
    name = "".join(content[1:21]).strip()
    file = open( name+ ".txt","r+",encoding="latin-1")
    file.seek(0,2)
    file.write("0")
    file.write(chr(page+1))
    file.write(chr(0))
    file.write(chr(0))
    number = math.floor((2042/ord(typeInfo[-1])))
```

```
    file.write(intToChar(number,2))
    for i in range(2042):
        file.write(chr(0))
    file.close()
```

c) Record Header and Record Structure:

Record Header	Rest of the Record			
1 byte	4 bytes	4 bytes		4 bytes
Deleted	Field_1	Field_2		Field_n
1-0	Data 1	Data 2	...	Data n

```
def createRecord():

    print("Type of the record that will be created: ")
    typeName = input()
    fields = []
    record = ["0"]
    page = 1
    full = False
    typeInfo = findType(typeName)


    if isinstance(typeInfo, str) :
            print("There is no such type in System Catalog. You can one
create if you want.")
        return
    numOfFields = ord(typeInfo[21])
    recordSize= ord(typeInfo[-1])
    if recordSize == 10:
        recordSize =13
```

```
    print("Values of Fields ( min 0 -- max 2^32-1):")
    for i in range(numOfFields):
        print("Field " + "".join(typeInfo[20*i+22:20*i+42]).strip() +" :
")
        while True:
            fieldVal = int(input())
            if (fieldVal) < 0 or (fieldVal) > math.pow(2,32):
                print("Invalid value, please type  min 0 -- max 2^32-1")
            else:
                fields.append(intToChar(fieldVal,4))
                break
    if findRecord(typeInfo,fields[0]) == "error":
        print("There is a Record with this key value, first field should
be unique")
        return

    record.extend(fields)
    ...
```

IV) Operations:

a) Create a type:

```
def createType():

    file = open("Sys_cat.txt","r+",encoding="latin-1")
    print("Name of the type: ( min 8 character -- max 20 character ): ")
    while True:
        typeName = input()
        if len(typeName) < 8 or len(typeName) >20:
            print("Invalid name, please type min 8 character -- max 20
character")
        else:
            break
    control = findType(typeName)
    if isinstance(control, list) :
        print("There is a type already with same name,Type Name should
be unique")
        return
    print("Number of Fields ( min 3 -- max 32):")
```

```
    while True:
        numOfFields = int(input())
        if numOfFields < 3 or numOfFields > 32:
            print("Please type a valid number: ( min 3 -- max 32)")
        else:
            break
    namesOfFields = []
    print("Please type field names one by one: ")
    for i in range(numOfFields):
        print("Field " + str(i+1) +" : ")
        while True:
            fieldName = input()
            if len(fieldName) < 8 or len(fieldName) >20:
                 print("Invalid name, please type min 8 character -- max
20 character")
            else:
                namesOfFields.append(fieldName)
                break

    newType = makeType(typeName,numOfFields, namesOfFields)

    page = 1
    content = diskDrive("Sys_cat",page)
    while True:
        pos = 0
        while pos + 663 < 2048:
            if content[pos] != chr(0):
                pos += 663
            else:
                content[pos:pos+663] = list(newType)
                file.seek((page-1)*2048)
                file.write("".join(content))
                file.close()
                createFile(newType)
                print("Type created successfuly")
                return
        page += 1
        content = diskDrive("Sys_cat",page)

def makeType(typeName,numOfFields,namesOfFields):
```

```
    myType = "0"

    for i in range(20 - len(typeName)):
        typeName += " "
    myType += typeName + chr(numOfFields)

    for i in range(32):
        if i < numOfFields:
            myType += namesOfFields[i]
            for k in range(20-len(namesOfFields[i])):
                myType += " "

        else:
            for k in range(20):
                myType += " "
    myType += chr(4*numOfFields+1)

    return myType
def createFile(typeInfo):
    content = list(typeInfo)
    name = "".join(content[1:21]).strip()
    file = open( name+ ".txt","w",encoding="latin-1")
    file.write("0")
    file.write(chr(2))
    file.write(chr(0))
    file.write(chr(0))
    number = math.floor((2042/ord(typeInfo[-1])))
    file.write(intToChar(number,2))
    for i in range(2042):
        file.write(chr(0))
    file.close()
```

b) Delete a type:

```
def deleteType():
    file = open("Sys_cat.txt","r+",encoding="latin-1")
    size = os.stat("Sys_cat.txt").st_size
    if size == 0:
        print("There is no type in System Catalog")
```

```
        return
    print("Name of the type that deleted: ( min 8 character -- max 20
character ): ")
    while True:
        nameToDelete = input()
        if len(nameToDelete) < 8 or len(nameToDelete) >20:
             print("Invalid name, please type min 8 character -- max 20
character")
        else:
            break
    page = 1
    content = diskDrive("Sys_cat",page)
    while True:
        pos = 0
        while pos +663 < 2048:
            if content[pos] == "0":
                name = "".join(content[pos+1:pos+21]).strip()
                if name == nameToDelete:
                    content[pos] = "1"
                    os.remove(name + ".txt")
                    print("type deleted: " ,name)
                    file.seek((page-1)*2048)
                    file.write("".join(content))
                    file.close()
                    return
                else:
                    pos += 663
            elif content[pos] == "1":
                pos += 663
            else:
                print("There exist no type as your type")
                file.close()
                return

        page += 1
        content = diskDrive("Sys_cat",page)
```

c) List all types:

```
def listTypes():
    empty = True
    size = os.stat("Sys_cat.txt").st_size
    if size == 0:
        print("There is no type in System Catalog")
        return

    page = 1
    content = diskDrive("Sys_cat",page)
    while True:
        pos = 0
        while pos +663 < 2048:
            if content[pos] == "0":
                name = "".join(content[pos+1:pos+21]).strip()
                numOfFields = ord(content[pos+21])
                namesOfFields=[]
                for i in range(numOfFields):

namesOfFields.append("".join(content[pos+20*i+22:pos+20*i+42]).strip())
                empty = False
                      print("Type name:", name,", Number of Fields: ",
numOfFields, ", Field Names: ", namesOfFields)
                pos += 663
            elif content[pos] == "1":
                pos += 663
            else:
                if empty:
                    print("There exist no type as your type")
                return
        page += 1
        content = diskDrive("Sys_cat",page)
```

d) Create a record:

```
def createRecord():

    print("Type of the record that will be created: ")
    typeName = input()
    fields = []
```

```
    record = ["0"]
    page = 1
    full = False
    typeInfo = findType(typeName)


    if isinstance(typeInfo, str) :
            print("There is no such type in System Catalog. You can one
create if you want.")
            return
    numOfFields = ord(typeInfo[21])
    recordSize= ord(typeInfo[-1])
    if recordSize == 10:
        recordSize =13

    print("Values of Fields ( min 0 -- max 2^32-1):")
    for i in range(numOfFields):
         print("Field " + "".join(typeInfo[20*i+22:20*i+42]).strip() +" :
")
        while True:
            fieldVal = int(input())
            if (fieldVal) < 0 or (fieldVal) > math.pow(2,32):
                print("Invalid value, please type  min 0 -- max 2^32-1")
            else:
                fields.append(intToChar(fieldVal,4))
                break
    if findRecord(typeInfo,fields[0]) == "error":
         print("There is a Record with this key value, first field should
be unique")
        return

    record.extend(fields)


    content = diskDrive(typeName,1)

    with open(typeName +".txt","r+", encoding='Latin-1') as file:
        while(True):
            if content[0] == "1":
                page += 1
```

```
                content = diskDrive(typeName,ord(content[1]))
                continue

            position = 6
            numberOfRecords = int(charToInt("".join(content[2:4])))
            count = 0

            while count < numberOfRecords:

                if content[position] == "1":
                    content[position : position + recordSize] = record
                    newNum = (intToChar(numberOfRecords+1,2))
                    if newNum == "".join(content[4:6]):
                        content[0] = "1"
                        full = True

                    content[2] = newNum[0]
                    content[3] = newNum[1]
                    file.seek((page-1)*2048)
                    file.write("".join(content))
                    if full:
                        createNewPage(typeInfo,ord(content[1]))
                    print("Type created successfuly")
                    return
                else:
                    position += recordSize
                    count += 1


content[numberOfRecords*recordSize+6:(numberOfRecords+1)*recordSize+6]
= record
            newNum = (intToChar(numberOfRecords+1,2))
            content[2] = newNum[0]
            content[3] = newNum[1]
            file.seek((page-1)*2048)
            file.write("".join(content))
            print("Type created successfuly")
            return
```

```
def findType(typeName):
    size = os.stat("Sys_cat.txt").st_size
    if size == 0:
        return "type error"

    page = 1
    content = diskDrive("Sys_cat",page)

    while True:
        pos = 0
        while pos +663 < 2048:
            if content[pos] == "0":
                name = "".join(content[pos+1:pos+21]).strip()
                if name == typeName:
                    tempList = content[pos:pos+663]
                    return tempList
                else:
                    pos += 663
            elif content[pos] == "1":
                pos += 663
            else:
                return "type error"

        page += 1
        content = diskDrive("Sys_cat",page)
```

e) Delete a record:

```
def deleteRecord():
    print("Type of the record that will be deleted: ")
    typeName = input()
    typeInfo = findType(typeName)
    page = 1

    if isinstance(typeInfo, str) :
        print("There is no such type in System Catalog")
        return
```

```
    recordSize= ord(typeInfo[-1])
    if recordSize == 10:
        recordSize =13
    print("Key value of the record that will be deleted: ")
    while True:
            keyVal = int(input())
            if keyVal < 0 or keyVal > math.pow(2,32):
                print("Invalid value, please type  min 0 -- max 2^32-1")
            else:
                break

    content = diskDrive(typeName,1)

    file = open(typeName +".txt","r+", encoding='Latin-1')

    while(True):
        position = 6
        numberOfRecords = int(charToInt("".join(content[2:4])))

        count = 0

        while count < numberOfRecords:
            if content[position] == "1":
                position += recordSize
            else:
                                                                temp     =
charToInt("".join(content[position+1:position+5]))

                if temp == keyVal:
                    content[position] = "1"
                    print("Record deleted successfuly")

                    content[0] = "0"
                    newNum = (intToChar(numberOfRecords-1,2))
                    content[2] = newNum[0]
                    content[3] = newNum[1]

                    file.seek((page-1)*2048)
                    file.write("".join(content))
                    file.close()
```

```
                    return
                else:
                    position += recordSize
                    count += 1

        if content[0] == "1":
            page += 1
            content = diskDrive(typeName,ord(content[1]))
        else:
            print("There is no such record")
            return
```

f) Search for a record:

```
def searchRecord():
    print("Type of the record that will be searched: ")
    typeName = input()
    typeInfo = findType(typeName)
    page = 1

    if isinstance(typeInfo, str) :
        print("There is no such type in System Catalog")
        return

    recordSize= ord(typeInfo[-1])
    if recordSize == 10:
        recordSize =13
    numOfFields = ord(typeInfo[21])
    print("Key value of the record that will be searched: ")
    while True:
            keyVal = int(input())
            if keyVal < 0 or keyVal > math.pow(2,32):
                print("Invalid value, please type  min 0 -- max 2^32-1")
            else:
                break

    content = diskDrive(typeName,1)

    while(True):
        position = 6
```

```
        numberOfRecords = int(charToInt("".join(content[2:4])))
        count = 0
        while count < numberOfRecords:
            if content[position] == "1":
                    position += recordSize
            else:
                                                                temp     =
charToInt("".join(content[position+1:position+5]))
                if temp == keyVal:
                     result = "Field " + "".join(typeInfo[22:42]).strip()
+ " : " + str(temp)
                    for k in range(numOfFields-1):
                          result += ", Field "+ "".join(typeInfo[20*k+42:
20*k +62 ]).strip() + ": " + str(charToInt("".join(content[position +
4*k + 5: position + 4*k+9 ])))

                    print(result)
                    return
                else:
                    position += recordSize
                    count += 1
        if content[0] == "1":
            page += 1
            content = diskDrive(typeName,ord(content[1]))
        else:
            print("There is no such record")
            break
```

g) List all records of a type:

```
def listRecords():
    print("Type of the record that will be listed: ")
    typeName = input()
    typeInfo = findType(typeName)
    recordNum = 0
    page = 1
    empty = True

    if isinstance(typeInfo, str) :
        print("There is no such type in System Catalog")
```

```
        return

    recordSize= ord(typeInfo[-1])
    if recordSize == 10:
        recordSize =13
    numOfFields = ord(typeInfo[21])

    content = diskDrive(typeName,1)
    while(True):
        position = 6
        numberOfRecords = int(charToInt("".join(content[2:4])))
        count = 0
        while count < numberOfRecords:
            if content[position] == "1":
                position += recordSize
            else:
                empty = False
                recordNum += 1
                    result = "Record " + str(recordNum) + "::  Field " +
"".join(typeInfo[22:42]).strip()       +       "       :       "       +
str(charToInt("".join(content[position+1:position+5])))
                for k in range(numOfFields-1):
                     result += ", Field "+ "".join(typeInfo[20*k+42: 20*k
+62 ]).strip()  + ": " + str(charToInt("".join(content[position + 4*k +
5: position + 4*k+9 ])))

                print(result)
                position += recordSize
                count += 1
        if content[0] == "1":
            page += 1
            content = diskDrive(typeName,ord(content[1]))
        else:
            if empty:
                print("There is no Record yet")
            break
```

h) Helpers

```
def diskDrive(fileName,pageNum):
    file = open(fileName +".txt","r",encoding="latin-1")
    base = (pageNum-1)*2048
    c = file.read()
    content = list(c)
    while len(content)-base < 2048:
        content.append(chr(0))
    page = content[base:base+2048]
    file.close()
    return page

def garbageCollector():
    file = open("Sys_cat.txt","r+",encoding="latin-1")
    c = file.read()
    content = list(c)
    newContent = []
    if len(c) ==0:
        return
    position = 0
    while position < len(c):
        if content[position] == "1":
            position += 663
            continue
        else:
            newlist = content[position: position+663]
            newContent.extend(newlist)

        position += 663

    file.seek(0)
    file.truncate()
    file.write("".join(newContent))
    file.close()

def intToChar(number,digit):
    result = []
    resString = ""
    count = 0
    while number > 0:
        result.insert(0,chr(number % 256))
```

```
        number = math.floor(number / 256)
        count += 1
    if count < digit:
        for i in range(digit - count):
            resString += chr(0)

    return resString + "".join(result)

def charToInt(myString):
    result =0
    count = 1
    for char in myString:
        result += math.pow(256,len(myString)-count)*ord(char)
        count += 1
    return int(result)

def findRecord(typeInfo,keyVal):
    page = 1

    typeName = "".join(typeInfo[1:21]).strip()
    recordSize= ord(typeInfo[-1])

    content = diskDrive(typeName,1)

    while(True):
        position = 6
        numberOfRecords = int(charToInt("".join(content[2:4])))
        count = 0
        while count < numberOfRecords:
            if content[position] == "1":
                    position += recordSize
            else:
                temp = "".join(content[position+1:position+5])
                if temp == keyVal:
                    return "error"
                else:
                    position += recordSize
                    count += 1
        if content[0] == "1":
            page += 1
```

```
            content = diskDrive(typeName,ord(content[1]))
        else:
            return "create"

def makeType(typeName,numOfFields,namesOfFields):
    myType = "0"

    for i in range(20 - len(typeName)):
        typeName += " "
    myType += typeName + chr(numOfFields)

    for i in range(32):
        if i < numOfFields:
            myType += namesOfFields[i]
            for k in range(20-len(namesOfFields[i])):
                myType += " "

        else:
            for k in range(20):
                myType += " "
    myType += chr(4*numOfFields+1)

    return myType

def createNewPage(typeInfo,page):

    content = list(typeInfo)
    name = "".join(content[1:21]).strip()
    file = open( name+ ".txt","r+",encoding="latin-1")
    file.seek(0,2)
    file.write("0")
    file.write(chr(page+1))
    file.write(chr(0))
    file.write(chr(0))
    number = math.floor((2042/ord(typeInfo[-1])))
    file.write(intToChar(number,2))
    for i in range(2042):
        file.write(chr(0))
    file.close()
```

```
def createFile(typeInfo):
    content = list(typeInfo)
    name = "".join(content[1:21]).strip()
    file = open( name+ ".txt","w",encoding="latin-1")
    file.write("0")
    file.write(chr(2))
    file.write(chr(0))
    file.write(chr(0))
    number = math.floor((2042/ord(typeInfo[-1])))
    file.write(intToChar(number,2))
    for i in range(2042):
        file.write(chr(0))
    file.close()

def findType(typeName):
    size = os.stat("Sys_cat.txt").st_size
    if size == 0:
        return "type error"

    page = 1
    content = diskDrive("Sys_cat",page)

    while True:
        pos = 0
        while pos +663 < 2048:
            if content[pos] == "0":
                name = "".join(content[pos+1:pos+21]).strip()
                if name == typeName:
                    tempList = content[pos:pos+663]
                    return tempList
                else:
                    pos += 663
            elif content[pos] == "1":
                pos += 663
            else:
                return "type error"

        page += 1
        content = diskDrive("Sys_cat",page)
```

V) Readme :

You can run .py code with python3 in any shell

python3 database.py

If you want you can change mode of the file as executable and run it directly in linux environment

chmod +x ./database.py
./database.py

Program is easy to use and output is so clear from command line.
System Catalogue's name is **"Sys_cat.txt"**

VI) Conclusions & Assessment:

I try to implement this project as realistic as possible and I create files in that manner. I stick to the first plan of my project very well and implement as the same as I design.

I think my code is good enough and run in real situations.

I also add some kind of garbage collector to deletion processes which the garbage collector is responsible for.

Page struct is coded well and working smoothly both in system catalog and in record files.

Hazar Çakır