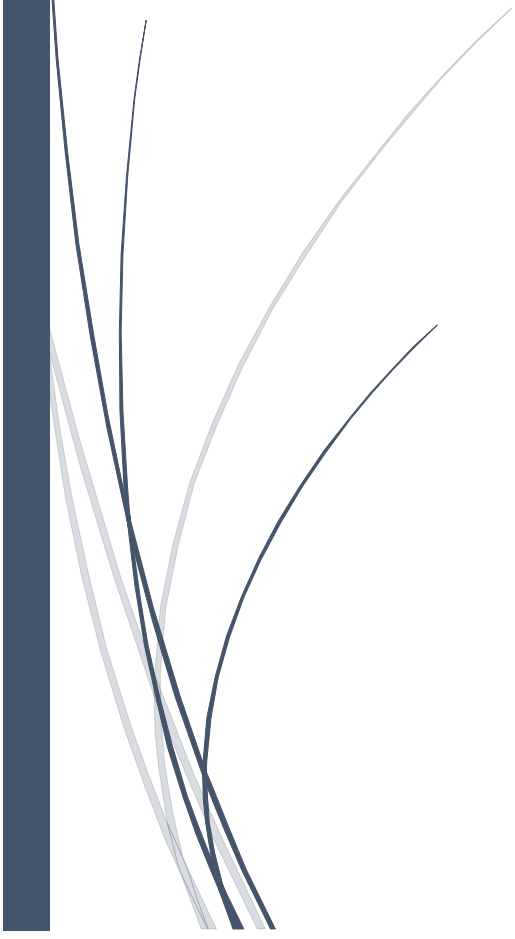
CMPE 321

Assignment 2

Implementing Storage Manager System

Cemal Aytekin - 2015400126



Contents

1. INTRODUCTION

A storage manager is a program that controls how the memory will be used to save data to increase the efficiency of a system. In the previous project, I had designed a Storage Manager System without error checking. While I was implementing the design, I had to change some a little bit the structures of the headers. This document consists of the implementation details for a simple database management system, including changes from previous project, sample outputs screenshots. I implemented it in Java.

2. CHANGES

2.1. System Catalogue

I changed the System Catalog Header as the following:

- Page Header(17 bytes)
  - Type Name (12 bytes)
  - Number of type system catalog includes (4 bytes)
  - isDeleted(1 byte)

2.2. Data Files

I changed the Data Files Pages Header as the following:

- Page Header (Number of Fields*12+8 bytes)
  - Number of Records (4 bytes)
  - Number of Fields (4 bytes)
  - Field Names (Number of Fields*12 bytes)

2.3 Pages

Now, in the page headers there is no pointer to next page and isEmpty. It includes only number of records it includes and page ids.

Also in the previous assignment, I sad that a page can include 40 records at most. Now I changed this assumption as the following: A page can include at most 10 records.

3. SAMPLE SCREENSHOTS

1.Menu

```
Welcome to DB Manager System!

..:: DB MENU  ::..

Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation:
```

2.Create a Type

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 1
type name: Student

Number of fields: 3

Field names:

field name #1: Name
field name #2: Surname
field name #3: Department
Student has been created successfuly

Press 1 to go back to Main Menu:
```

3. List All Types

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 3

3 type found!

        1. STUDENT: Name, Surname, Department

        2. TEACHER: Name, Surname, Age, Gender, Department

        3. PERSONEL: Name, Surname, Age, Payment, Job, Gender


Press 1 to go back to Main Menu:
```

4. Delete a Type

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 2

type name: Student
Student: Type is deleted!


Press 1 to go back to Main Menu: 1
```

5. Create a Record

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 4
type name:product
Enter record id: 17

cost: 186
weight: 123

Reading page #1 ...
Reading page #2 ...

SUCCESS! The record has been inserted successfuly to the page #2

Press 1 to go back to Main Menu:
```

6. Delete a Record

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 5

type name:product

record ID: 17

Reading page #1 ...
Reading page #2 ...
SUCCESS! The record deleted successfuly from the page #2


Press 1 to go back to Main Menu:
```

7. List Records

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 6

type name:Tools

Reading page #1 ...

        1. recordID: 1
                number: 4
                size: 16
                cost: 47

        2. recordID: 2
                number: 9
                size: 14
                cost: 56

        3. recordID: 6
                number: 12
                size: 98
                cost: 130

        4. recordID: 7
                number: 3
                size: 14
                cost: 3


Press 1 to go back to Main Menu:
```

8. Search a Record

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 7

type name:product

record ID: 3
Reading page #1 ...
        recordID: 3
                cost: 133
                weight: 3

Record has been found in the page #1!


Press 1 to go back to Main Menu:
```

9. Existence Error

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 1
type name: Student
ERROR: Type already exists
Enter another type name:
```

10. Exceeding Error

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 1
type name: Multitransistors
ERROR: Type names cannot be more than 12 characters long
Enter another type name:
```

11. No Such Record Found Error

```
Type Operations
(1) Create Type
(2) Delete Type
(3) List Types

Record Operations
(4) Insert a Record
(5) Delete a Record
(6) List all Records
(7) Search for a Record
(0) Exit

Please select an operation: 7

type name:product

record ID: 19
Reading page #1 ...
Reading page #2 ...


No such record found!

Press 1 to go back to Main Menu:
```

4. CONCLUSION & ASSESSMENT

In this experience, I see design and implementation always don't fit perfectly. I need to change some data structures. In this documentation a storage manager design is proposed where size of each structure is fixed. This creates an inefficiency in terms of memory usage while it makes the storage manager easier to implement. Also note that, the pages and records are inserted to storage manager linearly without any specific order. This makes searching slower whereas it makes insertion faster. It's faster when listing records. This restricts the user to some extent, however it makes the storage manager faster since the length controls are unnecessary. To sum up, this implementation has its own ups and downs just like every implementation. Since it is kept as a simple one, it is easy to modify and improve. Hence, implementing it would also be easier with necessary modifications that can be realized on the run.