BOĞAZİÇİ UNIVERSITY

CMPE 321

PROJECT 2

Summer 2017

Storage Manager Implementation

Mustafa Enes ÇAKIR

July 27, 2017

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, some things aren't fit to blueprints. I changed some data structures. 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 from the Initial Design

I add some attributes to data structures for accessing easily. Furthermore, I add a *Disc Directory* page for simulating virtual hard disc files. This design contains three main components which are Disc Directory, System Catalogue and Data Files.

2.1 Disc Directory

Disc directory is responsible for storing the file address. Also it keeps last free address.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Table 1: Hard Disc Abstraction

File Name	Address
Free Space	5
SysCat.txt	1
Dog.txt	2
Cat.txt	4

Table 2: Disc Directory

2.2 System Catalogue

System catalogue is responsible for storing the metadata. It's a blueprint for datas. It has multiple pages. Each record has a fixed size. I add two more fields to page headers and one more to record header. Each type has 10 fields. Therefore, the space of each record in the system catalog is 134 bytes. So 10 data type records can be stored in a page.

- Page Header (13 bytes)
  - Page ID (4 bytes)
  - Pointer to Next Page (4 bytes)
  - # of Records (4 bytes)
  - isEmpty (1 byte)
- Record (133 bytes)
  - Record Header (14 bytes)
    - Type Name (12 bytes)
    - # of Fields (1 byte)
    - isEmpty (1 byte)
  - Field Names (10 x 12 = 120 bytes)

Page ID	Pointer to Next Page		# of Records		isEmpty
Type Name 1	# of Fields	isEmpty	Field Name 1	...	Field Name 10
Type Name 2	# of Fields	isEmpty	Field Name 1	...	Field Name 10
...	...	...	...	...	...
Type Name 10	# of Fields	isEmpty	Field Name 1	...	Field Name 10

Table 3: a Page of a System Catalogue *(Starting with the Page Header)*

2.3 Data Files

Data files store actual datas. I didn'd change my data files. In this storage manager system, data files are separated into the number of types. Each data file can store one type of record and is named with that type. *"hamster.txt"* can store only the records of type *hamster* for example. These records can be thought as instances of a type in the system catalog. Therefore the fields

of the records are actual values. Here, all the values are integer. A record in a data file has a space of 45 bytes. Each page in a data file can store 30 records at most.

2.3.1 Pages

Page headers store information about the specific page it belongs to.

- Page Header (13 bytes)
  - Page ID (4 bytes)
  - Pointer to Next Page (4 bytes)
  - # of Records (4 bytes)
  - isEmpty (1 byte)
- Records (a Record = 45 bytes)

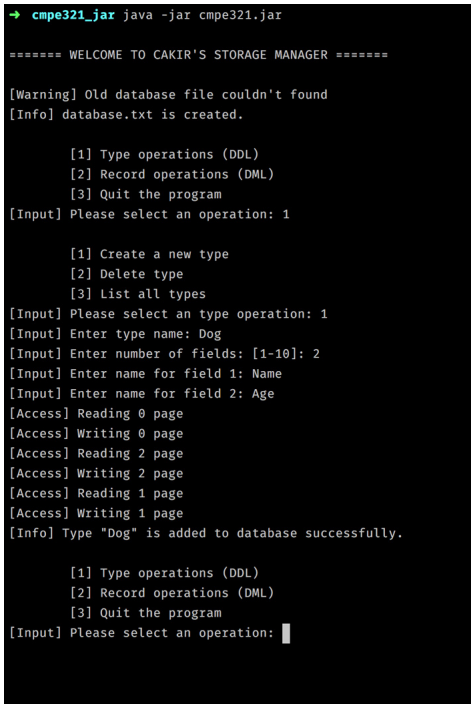
2.3.2 Records

- Record Header (5 bytes)
  - Record ID (4 bytes)
  - isEmpty (1 bytes)
- Fields (10 x 4 = 40 bytes)

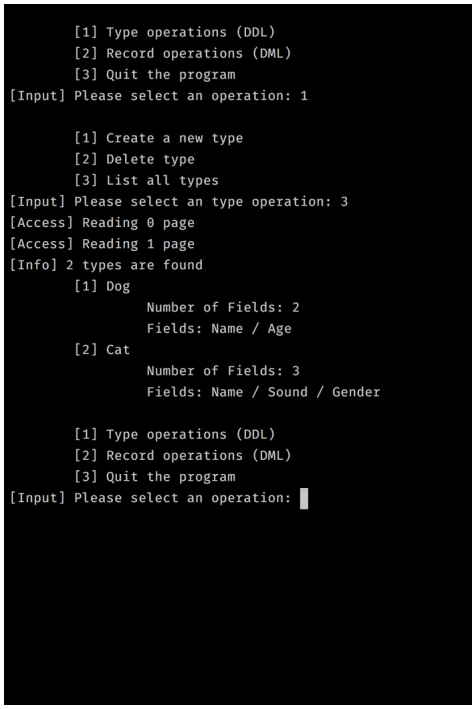
Page ID	Pointer to Next Page		# of Records		isEmpty
Record ID 1	isEmpty	Field 1	Field 2	...	Field 10
Record ID 2	isEmpty	Field 1	Field 2	...	Field 10
...	...	...	...	...	...
Record ID 30	isEmpty	Field 1	Field 2	...	Field 10

Table 4: Page of a Data File *(Starting with the Page Header)*

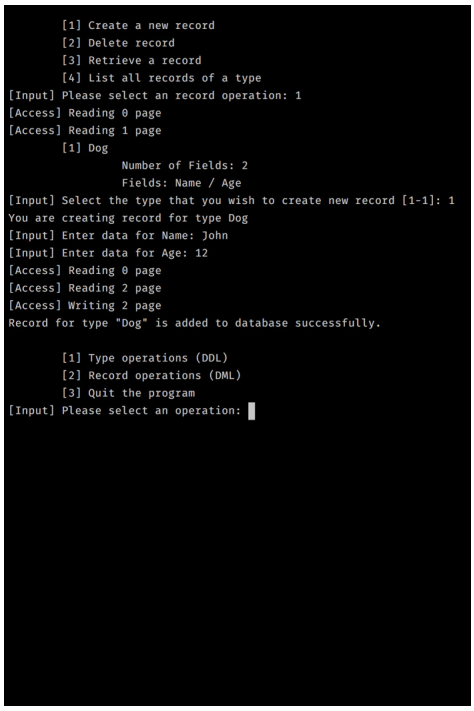
3 Sample Outputs



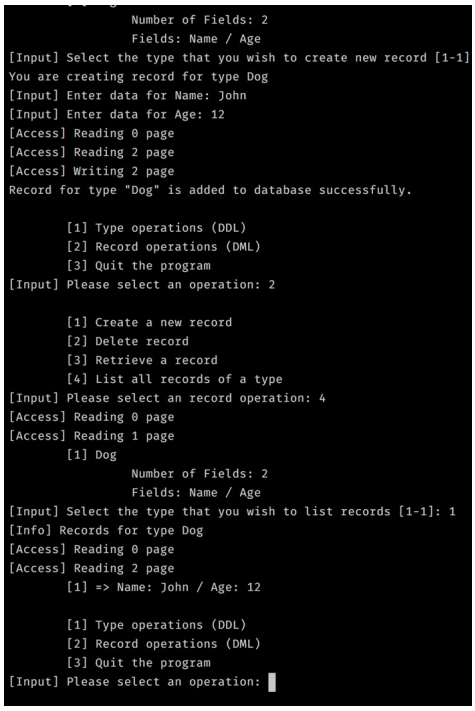
(a) Create type



(b) List all types



(a) Create record



(b) List all records of a type

4 Conclusions & 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. I allocated 1400 Bytes space for pages however the system only uses 1363 Bytes, this is done for Reliability. 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, no checking for identical key is needed.

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.