Spring 2018

CMPE 321

Assignment I

Designing 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. This project is about design of a storage manager system which supports a common data handling operation of DDL and DML.

a. **DDL Operations**
    1. Create a type
    2. Delete a type
    3. List all types
b. **DML Operations**
    1. Create a record
    2. Delete a record
    3. Search for a record by primary key
    4. List all records of a type

The design has a system catalog which stores metadata, and multiple data files that store the actual data. Via that manager, one can create new types, delete types and list all types. One can do also some record operations like creating a record, deleting a record, searching for a record and list all records of a type. This document includes some assumptions.

2. ASSUMPTIONS

- **Type**
    - Name of the data types contain only alphanumeric characters.
    - Type names are at most 12 characters long.
    - Duplicate names of data types are not allowed.
    - Data file names are as follows: “typeName.txt”.
    - A file has at most 100 pages.
    - UTF-8 standard is used in the system, it means a char equal 1 byte.
- **System Catalogue**
    - System catalogue file name is “syscat.txt”.
    - System catalogue file cannot be deleted by any user.
    - A page in System catalogue has at most 15 records.
- **Page**
    - Page Size is 1550 Bytes.
    - A page will have at most 40 records.
    - A page will contain at most one type of record.
- **Record**
    - A record can only be stored in 1 page
    - Records consist of 8 fields.
- **Field**
    - Field names are at most 12 characters long.
    - Duplicate names of fields of a data are not allowed.
    - All fields shall be of type integer.

3. DATA STRUCTURES

This design contains two main components which are System Catalogue and Data Files.

3.1. System Catalogue

The first component is System Catalogue which is very important in Database Management Systems. It is responsible for storing the metadata.

- Pointer to First Page (4 bytes)
- Page Header (5 bytes)
  - Page ID (4 bytes)
  - Number of Records (1 byte)
- Record (109 bytes)
  - Record Header (13 bytes)
    * Type Name (12 bytes)
    * Number of Fields (1 bytes)
  - Field Names (8 x 12 = 96 bytes)

Page Id			Number of Records		
Type Name 1	Number of Fields	Field Name 1	Field Name 2	...	Field Name 8
Type Name 2	Number of Fields	Field Name 1	Field Name 2	...	Field Name 8
Type Name 3	Number of Fields	Field Name 1	Field Name 2	...	Field Name 8
...	...	...	...	...	...
...	...	...	...	...	...
Type Name 15	Number of Fields	Field Name 1	Field Name 2	...	Field Name 8

Table 1. Page of a System Catalogue (Starting with the Page Header)

3.2. Data Files

The other component is data files which are responsible for storing the actual data in structured manner.

- File Header (5 bytes)
  - Pointer to First Page (4 bytes)
  - Number of Pages (1 bytes)

Pointer to First Page	Number of Pages
Page 1	
Page 2	
Page 3	
...	
...	
Page 100	

Table 2. File of a type (Starting with the File Header)

3.2.1 Pages

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

- Page Header (10 bytes)
  - Page ID (4 bytes)
  - Pointer to Next Page (4 bytes)
  - Number of Records (1 byte)
  - isEmpty (1 byte)
- Records (37 bytes)

3.2.2 Records

- Record Header (5 types)
  - Record ID (4 bytes)
  - isEmpty (1 bytes)
- Fields (8 x 4 = 32 bytes)

Page ID	Pointer to Next Page		Number Of Records		isEmpty
Record ID 1	isEmpty	Field1	Field 2	...	Field 8
Record ID 2	isEmpty	Field1	Field 2	...	Field 8
Record ID 3	isEmpty	Field1	Field 2	...	Field 8
...	...	...	...	...	...
...	...	...	...	...	...
Record ID 40	isEmpty	Field1	Field 2	...	Field 8

Table 3. Page of a Data File (Starting with the Page Header).

4. ALGORITHMS

4.1. DDL Operations

Database Design Language (DDL) operations are generally is related to system catalogue. As we have declared.

4.1.1. Create a Type

```
declare dataType
typeName ← Get From User
numberOfFields ← Get From User
dataType.push(typeName, numberOfFields)
for 0 to numberOfFields do
    fieldName ← Get From User
    dataType.push(fieldName)
end
for numberOfFields to 8 do
    dataType.push(NULL)
```

```
end
file ← open("syscat.txt")
file.push(dataType)
createFile(typeName.txt)
```

4.1.2. Delete a Type

```
typeName ← Get From User
deleteFile(typeName.txt)
file ← open("syscat.txt")
declare currentPage
currentPage = file.firstPage
foreach page in file do
    foreach record in currentPage do
            if record.isEmpty == 0 and record.typeName == typeName then
                    record.isEmpty ← 1
                    break
            end
            else
                    currentPage = currentPage.next
    end
end
```

4.1.3. List all Types

```
file ← open("syscat.txt")
declare currentPage
currentPage = file.firstPage
foreach page in file do
    foreach record in currentPage do
            if record.isEmpty = 0 then
                    print(record.typeName)
            end
    end
end
```

4.2 DML Operations

Database Manipulation Language (DML) are generally is related to data files.

4.2.1 Create a Record

```
recordType ← Get From User
fieldNum ← getFieldNumber(recordType, ”syscat.txt”)
file ← open(recordType.txt)
declare currentPage
currentPage = file.firstPage
foreach page in file do
    if currentPage.numOfRecords == 40 then
        currentPage = currentPage.next
    end
    else
        break;
end
foreach record in currentPage do
    if record.isEmpty = 1 then
        record.isEmpty ← 0
        for i ← 0 to fieldNum do
            record[i + 2] ← Get From User
        end
        page.numberOfRecords++
        break
    end
end
```

4.2.2 Delete a Record

```
recordType ← Get From User
recordID ← Get From User
file ← open(recordType.txt)
declare currentPage
currentPage = file.firstPage
foreach page in file do
    foreach record in currentPage do
        if record.isEmpty = 0 and record.id = recordID then
            record.isEmpty ← 1
            page.numOfRecords--
```

```
                **break**
            **end**
            **else**
                currentPage = currentPage.next
    **end**
**end**
```

4.2.3 Search For a Record

```
recordType ← Get From User
recordID ← Get From User
file ← open(recordType.txt)
declare currentPage
currentPage = file.firstPage
foreach page in file do
    foreach record in currentPage do
        if record.isEmpty == 0 and record.id == recordID then
            return record
        end
        else
            currentPage = currentPage.next
    end
end
```

4.2.4 List All Records Of a Type

```
declare records
recordType ← Get From User
file ← open(recordType.txt)
declare currentPage
currentPage = file.firstPage
foreach page in file do
    foreach record in currentPage do
        if record.isEmpty == 0 then
            records.push(record)
        end
        currentPage = currentPage.next
    end
end
return records
```

5. CONCLUSION & ASSESSMENT

In this project, I can conclude that I get a feeling how database management systems can be designed. I am expected to design a storage manager system that all constraints, structures and algorithms up to us.

This manager uses syscat.txt as the catalog and some txt files to store the information of the types and size of each structure is fixed. Thus this creates some extra storage needs because the memory management cannot be done dynamically. However this constant structure allows us to build a faster system. The design of the system increases reading performance. The system can reach every record by using its page id and record id. The record id allows us pointing the record in the page easily. This creates an inefficiency in terms of memory usage whereas it makes the storage manager easier to implement.

I allocated 1550 Bytes space for pages however the system only uses 1490 Bytes, this is done for Reliability.

Additionally, as you can see, the pages and records are inserted into storage manager linearly without any specific order, which makes insertion faster whereas makes searching slower.

Briefly, this design has its pros and cons just like every design. Since it is considered as a simple one, it is easy to modify and improve.