CMPE 321

Project 1-Designing Storage Manager System

Hatice Melike Ecevit
2016400138
Spring 2018

1.Introduction

In this project, I am expected to design a storage manager system that supports DDL operations and DML. There should be a system catalogue which stores metadata and multiple data files that store the actual data. This document explains my design by showing my assumptions, constraints, data structures and explaining the algorithms behind the DDL and DML operations in pseudocode.

2.Assumptions and Constraints

1. There should be only one system catalogue file which has the name ‘SysCat.txt’.
2. The system shall not allow to create more than one system catalog file or delete an existing one.
3. Every character is 1 byte and every integer is 4 bytes.
4. All of the field values are integers.
5. A page will be 1500 bytes.
6. Field names are at most 10 characters long.
7. A page cannot hold records of 2 different types, it has to hold at most one type.
8. Two fields of a record cannot have the same name.
9. Data files shall have the format type_name.txt.
10. Field names are alphanumeric.
11. A record type name is at most 12 characters.
12. A data file can contain at most 5 pages.
13. A record type can contain at most 10 fields provided by the user. If it contains less, remaining fields considered as null.
14. A file has at most 100 pages.

3.Data Structures

This storage manager contains two components which are System Catalogue and Data Files.

3.1 System Catalogue

System catalogue is the main file of the store manager. It is responsible for storing the metadata. Any change that can be done in the system via this file. It has the name 'SysCat.txt'. It has multiple pages.

1.Page Header (8 bytes)

1.1. Page ID (4 bytes)

1.2. # of records (4 bytes)

2.Record (115 bytes)

2.1. Record Header

2.1.1. Type Name (10 bytes)

2.1.2. # of Fields (not null) (4 bytes)

2.1.3. Deletion Status (isDeleted) (1 byte)

2.2. Field Names (10 * 10 = 100 bytes)

	Page Header		**Page ID**	**# of records**		
Record Header	**TypeName_1**	**# of Fields_1**	**FieldName_1**	**FieldName_2**	**…**	**FieldName_10**
	TypeName_2	**# of Fields_2**	**FieldName_1**	**FieldName_2**	**…**	**FieldName_10**
	…	**…**	**…**	**…**	**…**	
	TypeName_n	**# of Fields_n**	**FieldName_1**	**FieldName_2**	**…**	**FieldName_10**

3.2 Data Files

Data files store actual datas. Each data file can store at most one type of record. Data files have the name type_name.txt. Each page in a data file can store at most 32 records.

3.2.1 Pages

3.2.1.1. Page Header (13 bytes)

3.2.1.1.1. Page ID (4 bytes)

3.2.1.1.2. # of Records (4 bytes)

3.2.1.1.3. isEmpty (1 byte)

3.2.1.1.4 Pointer to Next Page (4 bytes)

3.2.1.2. Records

3.2.2. Records (46 bytes)

3.2.2.1. Record Header (6 bytes)

3.2.2.1.1. Record ID (4 bytes)

3.2.2.1.2. isDeleted (1 byte)

3.2.2.1.3. isEmpty (1 byte)

3.2.2.2. Record Fields (10 * 4 = 40 bytes)

	Page Header	**Page ID**	**# Of Records**	**isEmpty**	**Pointer to next page**
RecordID_1	**isEmpty_1**	**Field_1**	**Field_2**	**…**	**Field_10**
RecordID_2	**isEmpty_2**	**Field_1**	**Field_2**	**…**	**Field_10**
…	**…**	**…**	**…**	**…**	**…**
RecordID_32	**isEmpty_32**	**Field_1**	**Field_2**	**…**	**Field_10**

4.Algorithms

1.Create a Type

```
function createType
declare recordType
recordType.name <– User Input
recordType.numberOfFields <– User Input
for integer i=0 to recordType.numberOfFields
        recordType.fields[i].name <– User Input
endfor
if recordType.numberOfField is smaller than 10
        for int i=recordType.numOfFields+1 to 10
                recordType.fields[1].name <– (NULL)
        endfor
endif
file <– open('SysCat.txt')
write file recordType
file.pageHeader.numberOfRecords++
createFile('recordType.name.txt')
endFunction
```

2. Delete a Type

```
function deleteType
recordTypeName <– User Input
file <–  findFile(recordsTypeName)
delete file
catalogue <– open('SysCat.txt')
for each page in catalogue
        for each record in page
                if record.typeName = recordTypeName
                record.isDeleted <– 1
                endif
        endfor
endfor
endFunction
```

3. List All Types

```
function listAllTypes
declare types
file <– open('SysCat.txt')
for each page in file
        for each record in page
                if record.isDeleted=0
                        types.push(record.typeName)
```

```
                endif
            endfor
        endfor
        return types
        endFunction
```

4. Create a Record

```
function createRecord
recordType <– User Input
file <– open('SysCat.txt')
numOfFields <– file.recordType.numberOfFields
recordFile <– open('recordType.txt')
for each page in recordFile
        if page.pageHeader.numberOfRecords < 32
        lastPage <– page
        endif
endfor
lastPage.pageHeader.numberOfRecords++
for each record in lastPage
        if record.isEmpty = 1
        for integer i=0 to numOfFields
                record.fields[i] <– User Input
        endfor
        record.isEmpty <– 0
        endif
endfor
endFunction
```

5. Delete a Record

```
function deleteRecord
recordType <– User Input
primaryKey <– User Input
file <– open(recordType.txt)
for each page in file
        for each record in page
                if record.isDeleted = 0 and record.id = primaryKey
                        page.pageHeader.numberOfRecords - -
                        record.isDeleted <– 1
                        record.isEmpty <– 1
                endif
        endfor
endfor
endFunction
```

6. Search For a Record

```
function searchRecord
declare searchedRecord
recordType <– User Input
primaryKey <– User Input
file <– open('recordType.txt')
for each page in file
        for each record in page
                if record.id = primaryKey and record.isDeleted = 0
```

```
            searchedRecord <– record
            endif
        endfor
endfor
return searchedRecord
endFunction
```

7. List All Records Of a Type

```
function listRecords
declare allRecords
recordType <– User Input
file <– open('recordType.txt')
for each page in file
        for each record in page
            if record.isDeleted = 0 and record.isEmpty = 0
                allRecords.push(record)
            endif
        endfor
endfor
return allRecords
endFunction
```

5.Conclusions & Assessment

In this project, I have designed a simple storage manager which has a system catalogue file and data files. In my design each file can hold at most one record type and can have at most 100 pages. This makes accessing a record with its primary key faster but insertion is slower since we have to access a specific page to insert a record. Since we didn't do any error checking, if a user enters a wrong input, this storage manager cannot handle it. Also because of fixed page structure we lose some memory that we might be able to use in storing more data.

To sum up, this is a really simple storage manager design and it has its own pros and cons. But mostly, it is very efficient while accessing a record but not so much while insertion. But we can modify this design and improve it.