CMPE321
Assignment 1

Cemal Burak Aygün
2014400072

2018 SPRING

Contents

1 INTRODUCTION

Data should be stored in disk in an organized way to ensure accessing and modifying data in a secure and convenient way. A storage manager is a system for that purpose. In this assignment, I designed a simple storage manager where users can create, list, search and delete data of several types specified by them. The system consists of a System Catalog which has the name SystemCatalog.txt and many Data Files which has the name in the form <typeName>.txt. System Catalog stores the general meta data about the system whereas a Data File stores pages of records (actual data). In my design DDL and DML operations are supported.

2 ASSUMPTIONS AND CONSTRAINTS

- A character is 1 byte.
- There is exactly one System Catalog file named "SystemCatalog.txt"
- There is a specific Data File for each distinct type. Name of a Data File is in the form of "<typeName>.txt"
- A Data File can contain many pages.
- Size of a page is 1024 bytes. (1 KB)
- All the records in a page are of the same type.
- A type can have at least 2, at most 8 fields which are specified by the user.
- The first field specified by the user becomes the primary key of that type.
- Primary key of a type cannot be changed.
- Type and field names are alphanumeric.
- Type and field names can be at most 16 characters long.
- All the fields are of type integer.
- Size of a type remains fixed after it is created.

3 DATA STRUCTURES

3.1 System Catalog

System Catalog stores the general metadata of the database system. In my design, System Catalog file stores the information of total type number in the first 4 bytes. After that, the information about types are stored consecutively. Each type information spans exactly 146 bytes and consists of the following sections:

- typeName (16 bytes): Name of the type.
- maxRecNumInPage (1 byte): How many records of this type a page can contain at most. (See 'Data Files' section for calculation of this value.)
- fieldNum (1 byte): Number of fields this type has.
- field<i>Name (16 bytes): Name of the ith field. ($1 \leq i \leq 8$)

All 8 'fieldName's are stored in System Catalog file for a type even if the type has less than 8 fields. Also, all of the name fields are appended with '_' character until they are 16 bytes (characters) long. This is done so that every type information spans exactly the same size (146 bytes). This property simplifies operations on System Catalog file.

When a type is deleted from System Catalog, the last type is copied to the position of the type which is to be deleted and **typeNum** is decreased by 1. So, in System Catalog, all the **typeNum** consecutive record types corresponds to a valid type information.

type_num					
typeName	maxRecNumInPage	fieldNum	field1Name	...	field8Name
...	...	...	...	...	...
typeName	maxRecNumInPage	fieldNum	field1Name	...	field8Name

Table 1: Structure of SystemCatalog File

3.2 Data Files

A Data File has the name <typeName>.txt and contains pages of records of exactly one type.

<page structure>	<page structure>	...	<page structure>
...	...	...	...
<page structure>	<page structure>	...	<page structure>

Table 2: Structure of a Data File

3.2.1 Page Structure

A page contains records of exactly one type. The size of a page is fixed (1024 bytes). Maximum number of records a page can store depends on the type of records it stores. Every page has a page header which stores some information about the page. A page header consists of the following sections:

- pageID (4 bytes): Unique ID of the page.
- ptrToNextPage (4 bytes): Points to the next page of the file.
- isEmpty (1 byte): A flag specifying whether there is enough space for a new record to be inserted into the page or not.
- recNum (1 byte): Number of records in the page.

When a record is deleted from a page, the last record in that page is copied to the position of the record which is to be deleted and **recNum** is decreased by 1. So, in a page, all the **recNum** consecutive records corresponds to a valid (not deleted) record.

pageID	ptrToNextPage	isEmpty	recNum
<record structure>	<record structure>	...	<record structure>
...	...	...	...
<record structure>	<record structure>	...	<record structure>

Table 3: Structure of a Page

3.2.2 Record Structure

A Data File can contain exactly one type of records. So, the type of a record can be determined from the name of the Data File in which the record is stored.
Every type has a fixed number of fields and every the of each field is fixed

(4 bytes integer). Hence, the size of a record is fixed and can be determined from the information of related type in System Catalog.
Furthermore, all the valid (not deleted) records are accumulated in the top portion of a page, and the number of valid records in a page is stored in the page header and the algorithm are designed so that they cannot reach a deleted record while reading records in a page. Hence, there is no need to keep extra information of whether or not a record is deleted.
Considering all the situations explained above, in my design there no need for a record header. So, a record only consists of its field values.

field1 value (4)	...	fieldN value (4)

Table 4: Structure of a Record $2 \leq N \leq 8$

4 OPERATIONS

Explanations of some functions used in pseudocodes below:

- **createPage(page)** creates a new page, links it to given **page** and returns the newly created page.
- **decreasetRecNum(page)** decreases the value recNum stored in the header of given **page** by 1.
- **decreaseTypeNum(SystemCatalogFile)** decreases the value typeNum stored in the first 4 bytes of System Catalog file by 1.
- **getFieldNum(recordType)** returns the number of fields given **recordType** has.
- **getRecNum(page)** returns the value recNum stored in the header of given **page**.
- **getRecord(page, i)** returns the ith record of given **page**.
- **getType(SystemCatalogFile, i)** returns the ith record type in System Catalog file.
- **getTypeNum(SystemCatalogFile** returns the value typeNum stored in the first 4 bytes of System Catalog File.
- **increasetRecNum(page)** increases the value recNum stored in the header of given **page** by 1.

- **increaseTypeNum(SystemCatalogFile)** increases the value typeNum stored in the first 4 bytes of System Catalog file by 1.
- **isEmpty(page)** returns the value isEmpty stored in the header of given **page**.
- **insertRecord(page, fieldValueArray)** inserts the values given in **fieldValueArray** to record position after the last record in given **page**.
- **insertType(SystemCatalogFile, typeName, fieldNameArray)** inserts given **typeName** and the values in **fieldNameArray** to end of System Catalog fie using the information in the first 4 bytes of the file.
- **updateRecord(page, i, record)** overwrites the position of the ith record in given **page** with given **record**.
- **updateType(SystemCatalogFile, i, recordType)** overwrites the position of the ith record type in System Catalog file with given **recordType**.

4.1 DDL Operations

4.1.1 Creating a Type

```
typeName ← USER_INPUT
numOfFields ← USER_INPUT
declare fieldNameArray
for i ← 1 to numOfFields do
  | fieldNameArray.push(USER_INPUT)
end
createFile(typeName.txt)
sysCat ← openFile(SystemCatalog.txt)
insertType(sysCat, typeName, fieldNameArray)
increaseTypeNum(sysCat)
```

4.1.2 Deleting a Type

```
typeNameToBeDeleted ← USER_INPUT
deleteFile(typeNameToBeDeleted.txt)
sysCat ← openFile(SystemCatalog.txt)
numOfTypes ← getTypeNum(sysCat)
for i ← 1 to numOfTypes) do
    recordType ← getType(sysCat, i)
    if recordType.typeName = typeNameToBeDeleted then
        if i ≠ numOfTypes then
            lastRecordType ← getRecordType(sysCat, numOfTypes)
            updateType(sysCat, i, lastRecordType)
        end
        decreaseTypeNum(sysCat)
        return
    end
end
```

4.1.3 Listing All Types

```
sysCat ← openFile(SystemCatalog.txt)
for i ← 1 to getTypeNum(sysCat) do
    recordType ← getType(sysCat, i)
    print(recordType)
end
```

4.2 DML Operations

4.2.1 Creating a Record

```
typeName ← USER_INPUT
sysCat ← openFile(SystemCatalog.txt)
recordType ← getType(sysCat, typeName)
numOfFields ← getFieldNum(recordType)
declare fieldValueArray
for i ← 1 to numOfFields do
    fieldValueArray.push(USER_INPUT)
end
dataFile ← openFile(typeName.txt)
declare page
foreach page in dataFile do
    if isEmpty(page) then
        insertRecord(page, fieldValueArray)
        increaseRecNum(page)
        return
    end
end
newPage ← createPage(page)
insertRecord(newPage, fieldValueArray)
increaseRecNum(newPage)
```

4.2.2 Deleting a Record

```
typeName ← USER_INPUT
primaryKey ← USER_INPUT
dataFile ← openFile(typeName.txt)
foreach page in dataFile do
    numOfRecords ← getRecNum(page)
    for i ← 1 to numOfRecords do
        record ← getRecord(page, i)
        if record.field1 = primaryKey then
            if i ≠ 1 then
                lastRecord ← getRecord(page, numOfRecords)
                updateRecord(page, i, lastRecord)
            end
            decreaseRecNum(page)
            return
        end
    end
end
```

4.2.3 Searching for a Record

```
typeName ← USER_INPUT
primaryKey ← USER_INPUT
dataFile ← openFile(typeName.txt)
foreach page in dataFile do
    numOfRecords ← getRecNum(page)
    for i ← 1 to numOfRecords do
        record ← getRecord(page, i)
        if record.field1 = primaryKey then
            return record
        end
    end
end
```

4.2.4 Listing All Records of a Type

```
typeName ← USER_INPUT
dataFile ← openFile(typeName.txt)
foreach page in dataFile do
    numOfRecords ← getRecNum(page)
    for i ← 1 to numOfRecords do
        record ← getRecord(page, i)
        print(record)
    end
end
```

5 CONCLUSIONS AND ASSESSMENT

In this design, a user can create new types which can have at least 2 and at most 8 fields. All the fields can be of type integer only and the first field is considered as the primary key.

I chose to store exactly one type of records in a Data File name of which is in the form "<typeName>.txt" in order to group records of the same type together.

In my design a page is 1024 bytes. Size of a record is dependent on the number of fields it has. So, the maximum number of records a page can store can be calculated by the following formula:

$$FLOOR((PageSize - PageHeaderSize)/RecordSize)$$

This causes some space in records to be wasted.

I keep all the structures fixed-sized in order to keep operations on files/pages easy. The biggest disadvantage of doing this is the unused space. For example, in System Catalog file, I create and store all the 8 fields of a type even if the type has less than 8 fields. Also, I append every type name and field name with '_' character to make them 16 bytes (charactes) long.

Since there is no structures such as indexes in my design, searching for a record is slow (linear). It simply goes to the Data File in which the record is stored and examines all of the pages in it one by one until the record is found.

Deleting a record can be considered as slow. When the record to be deleted is found in a page, the last record in that page is copied to the position of the record to be deleted and the value recNum in that page is decreased by one. This extra work ensures that all the consecutive recNum records in a page corresponds to valid (not deleted) records.

Since all the valid records accumulate at the top portion of a page, creating a record is fast. Storage Manager just finds the first empty page and inserts the record after the last one in that page. Then, increases the value recNum by 1.

Listing all the records of a type does not require the extra work for checking whether a record is valid (not deleted). Storage Manager just goes through the first recNum records in a page knowing that they are valid.

A similar system is used in System Catalog and DDL operations.

In conclusion, keeping all the structures fixed-sized and the design simple makes implementation, modification and maintenance easy at the cost of inefficiency in disk space usage and some operations such as searching.