CMPE321

Introduction to Database Systems

Storage Manager Implementation

Fatih İver

2016400264

Spring 2019

INTRODUCTION

This project aims to teach the internal dynamics of storage managers. Functionalities which are creating, deleting, and listing record types; creating, searching, updating, deleting and listing records are implemented to understand how fundamental operations work in a storage manager. Since implementing a storage manager requires a lot of design choices, designing such a system requires the preparer to think carefully and pay attention to petty details.

It is assumed that users will always give a valid input and disk manager exists.

Each character is 1 Byte – ASCII Character

ASSUMPTIONS & CONSTRAINTS

Type

- Encoding is Ascii
- Maximum number of characters allowed is 8 character for the type name and each field name
- Maximum number of fields allowed is 10 field per type – the name of the type is stored separately
- A type is stored as and in the given order:
    - Number of fields – 8 Bytes
    - Name of the type – 8 Bytes
    - Name of the fields – 10 * 8 Bytes = 80 Bytes
- A type occupies 96 Bytes of space on the disk

Record

- Encoding is Ascii
- All fields are integers and negative numbers are allowed
- Integers are stored as signed integers and by using 8 Bytes
- A record is stored as and in the given order:

  - A Boolean which shows a record is valid or not – 8 Bytes
  - An integer which is the key of the record – 8 Bytes
  - An integer which shows number of fields – 8 Bytes
  - 10 Integers which corresponds values of the fields – 80 Bytes
- A type occupies 104 Bytes of space on the disk

Type Page

- Page size is 2888 Bytes
- A page consists a page header and 30 types
- Page header is 8 Bytes
- Page header consists of number of records
- Number of records is 8 Bytes
- Each page consists 30 types
- A page is stored as and in the order:
  - Integer that shows the number of types – 8 Bytes
  - 30 Types – 30 * 96 = 2880 Bytes in Total

Record Page

- Page size is 3128 Bytes
- A page consists a page header and 30 record types
- Page header is 8 Bytes
- Page header consists of number of records
- Number of records is 8 Bytes
- Each page consists 30 record types
- A page is stored as and in the order:
  - Integer that shows the number of records – 8 Bytes
  - 30 Records – 30 * 104 = 3120 Bytes in Total

Type File

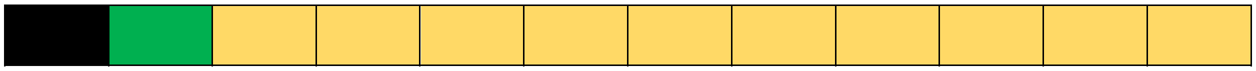
- Type file is the System Catalog file which is “sys.cat”
- “sys.cat” may contain infinitely many pages

Record File

- Each record file contains maximum 1000 record pages
- First file names as “<type_name>0.txt” and others file are named by incrementing the 0 at the end one by one.
- Full record file occupies 3128000 Bytes of space on the disk
- There is no limit on the number of pages a type can have – in other words, it solely depends on the capacity of the system

STORAGE STRUCTURES

Type

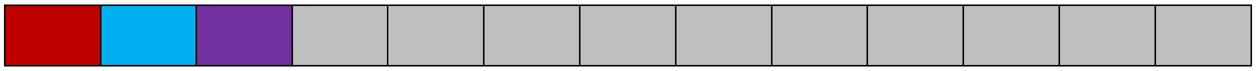


Black cell represents the number of valid fields which is 8 Bytes

Green cell represents the type name which is 8 Bytes

Yellow cells represent field names, each is 8 Bytes

Record



Red cell represents the validness of the record which is a Boolean value

Blue cell represents the key of the records which is an integer – 8 Bytes

Purple cell represents the number of valid fields in the record – 8 Bytes

Gray cells represent the field values of the record, each is 8 Bytes signed integers

OPERATIONS

Data Definition Language Operations

Create a Data Type

```
recordType = input()

recordFields = []

fieldNumber = int(input())

for i in range(fieldNumber):
    recordField = input()
    recordFields.append(recordField)

count = 0

Label: "MainLoop"
while True:
    file = open(“SysCat” + str(count))
    pageCount = 0
    foreach page in file:
        page.header.numberOfRecords != 15:
            foreach record in page:
                if record.type == 0:
                    record.type = recordType
                    i = 0
                    foreach field in recordFields:
                        record.field[i] = recordFields[i]
                        i += 1
                    page.header.numberOfRecords += 1
                    break "MainLoop"
        pageCount += 1
        if pageCount == 1000:
            break
    count += 1


createDataFile(recordType + str(0))
```

Delete a Data Type

```
recordType = input()
deleteFile(recordType)

count = 0

while isFileExist(recordType + str(count)):
    deleteFile(recordType + str(count))
    count += 1

count = 0
```

```

Label: "MainLoop"
while True:
    file = open("SysCat" + str(count))
    pageCount = 0
    foreach page in file:
        page.header.numberOfRecords != 0:
            foreach record in page:
                if record.type == recordType:
                    record.type = 0
                    page.header.numberOfRecords -= 1
                    break "MainLoop"
        pageCount += 1
        if pageCount == 1000:
            break
    count += 1
```

List All Types

```
count = 0

Label: "MainLoop"
while isFileExist("SysCat" + str(count)):
    file = open("SysCat" + str(count))
    pageCount = 0
    foreach page in file:
        page.header.numberOfRecords != 0:
            foreach record in page:
                if record.type != 0:
                    print(record.type)
        pageCount += 1
        if pageCount == 1000:
            break
    count += 1
```

Data Manipulation Language Operations

Create a Record

```
recordType = input()

count = 0

Label: "MainLoop"
while True:
    recordFile = open(recordType + str(count))
    pageCount = 0
    foreach page in recordFile:
        if page.header.numberOfRecords != 28:
            foreach record in page:
                if record.header.isEmpty:
                    record.header.isEmpty = False
                    fieldNumber = int(input())
                    for i in range(fieldNumber):
                        record.fields[i] = int(input())
                    page.numberOfRecords += 1
                    break "MainLoop"
        pageCount += 1
        if pageCount == 1000:
            break
    count += 1
```

Update a Record

```
recordType = input()

recordID = int(input())
record_fields = list(input())
count = 0

Label: "MainLoop"
while True:
    recordFile = open(recordType + str(count))
    pageCount = 0
    foreach page in recordFile:
        if page.header.numberOfRecords != 0:
            foreach record in page:
                if !record.header.isEmpty and record.ID == recordID:
                    record.fields = record_fields
                    break "MainLoop"
        pageCount += 1
        if pageCount == 1000:
            break
    count += 1
```

Delete a Record

```
recordType = input()

recordID = int(input())

count = 0

Label: "MainLoop"
while True:
    recordFile = open(recordType + str(count))
    pageCount = 0
    foreach page in recordFile:
        if page.header.numberOfRecords != 0:
            foreach record in page:
                if !record.header.isEmpty and record.ID == recordID:
                    record.ID = 0
                    page.header.numberOfRecords -= 1
                    break "MainLoop"
        pageCount += 1
        if pageCount == 1000:
            break
    count += 1
```

Search for a Record

```
recordType = input()
recordID = int(input())
```

```

count = 0

Label: "MainLoop"
while True:
    recordFile = open(recordType + str(count))
    pageCount = 0
    foreach page in recordFile:
        if page.header.numberOfRecords != 0:
            foreach record in page:
                if !record.header.isEmpty and record.ID == recordID:
                    return record
        pageCount += 1
        if pageCount == 1000:
            break
    count += 1
return None
```

List All Records of a Type

```
recordType = input()

count = 0

Label: "MainLoop"
while isFileExist(recordType + str(count)):
    recordFile = open(recordType + str(count))
    pageCount = 0
    foreach page in recordFile:
        if page.header.numberOfRecords != 0:
            foreach record in page:
                if !record.header.isEmpty:
                    print(record.type)
        pageCount += 1
        if pageCount == 1000:
            break
    count += 1
```

CONCLUSIONS & ASSESMENT

Implementation of a storage manager is showed in this project. Tough it is not a practical one, it is simple to understand the concepts in database management systems. It is not practical because this storage manager keeps 10 record fields for each record types and It does not matter whether a record type needs that much field. It is assumed that all record type will have 10 record fields even they don't need it, required space will be occupied on the disk. If it is not needed, they will be stored as empty string or 0. This approach is not efficient in terms of storage, but it makes the storage manager simple to implement. Another problem with this implementation is the limit on the number of characters for the field names. Each field name is maximum 8 Bytes, since ASCII characters are used, it allows maximum 8 characters naming which is ideal because average world length in English is 8 characters. However, it limits users in terms of naming with this character limitation. If users want support for the other languages in the world, it will require to change the whole structure of the system catalog file. Another problem is search times. No order is used for insertion, since first empty slot is used. This makes searching time linear. I think such systems should provide at least logarithmic time since this is a database management system and it is possible to have so many queries in short intervals. A sorting scheme may have been used but it would make the system more complex. This implementation is made for the sake of simplicity.