CMPE 321 Summer 2019
Project1-Designing Storage Manager System

Ege Onur Güleç - 2015300084

July 21, 2019

1 Introduction

In this assignment I want to design a storage manager system which consists of a system catalogue and some data files. Data will be stored in records which will be stored in pages and pages will be stored in files. I will make some assumptions and determine some limitations about inputs. Then I will give pseudo codes for DDL and DML operations.
First of all I will explain the assumptions then give the detailed pseudo codes.Following are the relevant DDL and DML operations that are available to user.
DDL Operations
1) Create a type
2) Delete a type
3) List all types
DML Operations
Create a record
Delete a record
Search for a record (by primary key)
Update a record (by primary key)
List all records of a type

1.1 Assumptions

Page size is 1KB
File size is 6KB
In each page there is a page header which has an unique page id, information about the remaining size in the page and a pointer to the next page
In each record there is a record header which has also an unique record id(primary key), info about whether record empty or not is and a pointer to the next record
Max number of fields a type can have is 10
Max length of a type name is 20
Max length of a field name is 16
User always enters valid input.
All fields shall be integers. However, type and field names shall be alphanumeric.
A disk manager already exists that is able to fetch the necessary pages when addressed.

2 Structure Design

We begin by creating the system catalogue.

number of types					
first type	number of files	number of pages	number of records	first field	...
second type	number of files	number of pages	number of records	first field	...
.....	....	.....	.....	.....	.....

Above table shows the design of system catalogue. It stores the metadata about the system

2.1 File

A file has multiple pages in it. When all the pages in a file are full we can create a new a file and a page of the desired type and add, update or delete records from that new file until its all pages are also full or deleted.

page1
page2
page3
....
....

Above table shows the structure of a file which can be created for any type

2.2 Page

A page consists of two different things. One of them is page header which includes an unique page id and a pointer to the next page and the other is several records which is another structure I will explain in the next section.

Page ID (4byte)
A pointer to the next page(8 byte)
Remaining size in the page(4byte)

Above table shows the structure of the page header

Page Header
Record 1
Record 2
...
...

Above table shows the structure of a page which can be created for any type

2.3 Record

A record consists of two different things. First one is the record header which has the record id, info about whether a record is empty or not and also a pointer to the next record. Second one is that a record has at most 10 fields store the relevant data for its specific type

Record ID (4byte)
isEmpty (1byte)
A pointer to the next record (8byte)

Above table shows the structure of the record header

Record Header
First Field
Second Field
....
....

Above table shows the structure of a record

3 Algorithms for Operations

First DDL operations:

Input: typeName,number of fields,names of fields
systemCatalog.open();
construct dataType;
while $i \leq numberofFields$ **do**
| dataType.fieldName[i]=namesofFields[i]
end
numberofTypes++
systemCatalogue.push(dataType);
systemCatalog.close();
$file \leftarrow createFile(typeName)$

Algorithm 1: Create a Type

Input: typeName
systemCatalog.open();
numberofTypes - -;
$k \leftarrow numberofFilesIntypeName$;
while $i \leq k$ **do**
  delete file named typeName[i];
end
delete line associated with typeName in system catalog;
systemCatalog.close();

Algorithm 2: Delete a Type

systemCatalog.open();
$k \leftarrow numberofTypes\ in\ systemcatalog$;
while $i \leq k$ **do**
  $j \leftarrow reads\ line[i]\ \ in\ \ systemcatalog$;
  print(j.typeName);
end
systemCatalog.close();

Algorithm 3: List all Types

```
Input: typeName,recordFields
systemCatalog.open();
k ← typeName.numberofFiles in systemcatalog ;
while i ≤ k do
    if typeName.file[i].remainingSize ≥ record.size then
        for all pages p in typeName.file[i] do
            p.remainingSize=p.header.readByte[12];
            if p.remainingSize ≥ record.size then
                construct new record r with recordFields;
                p.push(r);
                typeName.numberofRecords ++ ;
                systemCatalog.close();
                return r
            else
                next page;
            end
        end
    else
        next file;
    end
    end
    construct a new file typeName[k+1];
    construct a new page p;
    construct a new record r with recordFields;
    p.push(r);
    typeName[k+1].push(p);
    typeName.numberOfFiles++;
    typeName.numberofRecords ++ ;
    systemCatalog.close();
    return r
```

Algorithm 4: Create a record

```
Input: typeName,recordID
systemCatalog.open();
k ← typeName.numberofFiles in systemcatalog ;
while i ≤ k do
    for all pages p in typeName.file[i] do
        for all records r in p do
            r.recordID=r.header.readByte[0](read first 4 bytes of header);
            if r.recordID==recordID then
                delete r;
                if typeName.file[i].isEmpty==TRUE then
                    delete typeName.file[i];
                    rename other files accordingly;
                else
                    print("File is not empty")
                end
                typeName.numberofRecords - -;
                print("Record deleted successfully");
                systemCatalog.close();
                return
            else
                next record;
            end
        end
    end
end
print("Record does not exist");
systemCatalog.close();
return
```

Algorithm 5: Delete a Record

```
Input: typeName,recordID
systemCatalog.open();
k ← typeName.numberofFiles in systemcatalog ;
while i ≤ k do
    for all pages p in typeName.file[i] do
        for all records r in p do
            r.recordID=r.header.readByte[0](read first 4 bytes of header);
            if r.recordID==recordID then
                print("Record is found");
                return r ;
            else
                next record;
            end
        end
    end
end
print("Record does not exist");
systemCatalog.close();
return
```

Algorithm 6: Search a Record (by primary key)

```
Input: typeName,recordID,recordFields
systemCatalog.open();
k ← numberofFilesIntypeName ;
while i ≤ k do
    for all pages p in typeName.file[i] do
        for all records r in p do
            r.recordID=r.header.readByte[0](read first 4 bytes of header);
            if r.recordID==recordID then
                r.updateFields(recordFields) updating the given fields;
                print("Fields are updated");
                systemCatalog.close();
                return;
            else
                next record;
            end
        end
    end
end
print("Record does not exist");
systemCatalog.close();
return;
```

Algorithm 7: Update a Record(by primary key)

```
Input: typeName
systemCatalog.open();
k ← numberofFilesIntypeName ;
while i ≤ k do
    for all pages p in typeName.file[i] do
        for all records r in p do
            r.isEmpty=r.header.readByte[4](read byte 4 to 5 in header);
            if r.isEmpty==false then
                print(r);
            else
                next record ;
            end
        end
    end
end
systemCatalog.close();
```

Algorithm 8: List all Records

4 Conclusions and Assessment

In this project I had an intuition about how to design a database management storage. My designs advantages are since I insert records linearly without any specific order insertion is fast however, as a disadvantage searching is costly since it looks page by page and record by record.
Another advantage is that I use page ids and record ids they make easier to find a page in a file or find a record in a page but as a disadvantage it allocates more memory. This ids make easier to implement my design. Hopefully in the next project I will improve my design.