CmpE 321 Introduction to Database Systems
2018 Spring
Assignment 1
Designing Storage Manager System

Baran Kılıç
2014400123

March 12, 2018

1 Introduction

In this assignment, we are expected to design a simple storage manager. The storage manager have two parts system catalog and data files. Files consist of pages. Pages consist of records. Records consist of fields. We should be able to create, delete and see types with this storage manager. In addition, we should be able to create, delete, search and list records.

2 Assumptions and Constraints

- Page size is 1024 bytes.
- All fields shall be of type integer (4 bytes).
- Type names shall be alphanumeric.
- Field names shall be alphanumeric.
- Max length of a type name is 32 characters.
- Max length of a field name is 32 characters.
- The size of a character is 1 byte.
- Every type is stored in a different file.
- Max number of fields for a type is 10.
- Max file size is 10 kilobytes. (10 pages).
- System catalog header contains the number of types and the index of the next empty space for a type.
- The types contains full empty info, type name, id of the file that contains the record for that type, primary key bitmap (array of booleans, if the corresponding field is a primary key, its value is true) and field names.
- Page header contains the id of the next page, the number of records and the index of the next empty record.
- Record header contains the field full info. It is true if there is a record. It is false if it is deleted or emtpy.

3 Data Structures

3.1 System Catalog

Type count	Next empty index						} Header
Full empty info	Type name	File id	Primary key bitmap	Field name 1	...	Field name 10	} Types
Full empty info	Type name	File id	Primary key bitmap	Field name 1	...	Field name 10	
⋮							

3.2 File

Type name
Page 1
Page 2
⋮

3.3 Page

Page Header			**Remaining page**		
Next page id	Number of records	Next empty record index	Record 1	Record 2	...

3.4 Record

Record Header	**Remaining record**									
Full/empty info	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8	Field 9	Field 10

4 Operations

4.1 DDL Operations

4.1.1 Create a type

System catalog: syscat

```
function create_type(type_name, field_names[0:9],primary_keys[0:9])
    syscat.type_count <- syscat.type_count + 1
    index <- syscat.next_empty_index
    syscat.types[index].full <- true
    syscat.types[index].type_name <- type_name
    file_id <- createFile(type_name)
    syscat.types[index].file_id <- file_id
    syscat.types[index].primary_keys[0:9] <- primary_keys[0:9]
    syscat.types[index].field_names[0:9] <- field_names[0:9]

    // find next empty index
    while syscat.types[index].full = true do
        index <- index + 1
    endwhile
    syscat.next_empty_index <- index
end
```

4.1.2 Delete a type

```
function delete_type(type_name)
    // find the index of type to be deleted
    index <- 0
    while not (syscat.types[index].type_name = type_name) do
        index <- index + 1
    endwhile
```

```
    // delete it
    syscat.types[index].full <- false

    // update next empty index
    if index < syscat.next_empty_index then
        syscat.next_empty_index <- index
    endif
end
```

4.1.3 List all types

```
function list_types()
    index <- 0
    for i <- 0 to syscat.type_count - 1 do
        while syscat.types[index].full = false do
            index <- index + 1
        endwhile
        print( syscat.types[index].type_name)
    endfor
end
```

4.2 DML Operations

4.2.1 Create a record

```
function create_record(type_name,field_values[0:9])
    // find the index of type
    index <- 0
    while not (syscat.types[index].type_name = type_name) do
        index <- index + 1
    endwhile

    // get the file
    file_id <- syscat.types[index].file_id
    file <- getFile(file_id)

    // find a page to insert the record
    i <- 0
    while (i <  file.pages[].length) and (file.pages[i].next_empty_record_index = -1) do
        i <- i +1
    endwhile
    if i >=  file.pages[].length then
        error(file is full)
        return
    endif

    // add record
    page <- file.pages[i]
    record <- page.records[page.next_empty_record_index]
    record.full <- true
```

```
    record.fields[0:9] <- field_values[0:9]
    page.number_of_records <- page.number_of_records + 1

    // find next empty space
    index <- page.next_empty_record_index
    while (page.records[index].full = true) and (index < page.records[].length ) do
        index <- index +1
    endwhile
    if index >= page.records[].length then
        page.next_empty_record_index <- -1
    else
        page.next_empty_record_index  <- index
    endif
end
```

4.2.2 Delete a record

```
function delete_record(type_name,key_values[0:n])
    pos_string <- search_record(type_name,key_values[0:n])
    file_id <- substring(0, index_of_first_slash )
    page_id <- substring( index_of_first_slash,  index_of_second_slash)
    rec_index <- substring(index_of_second_slash)

    // access record
    file <- getFile(file_id)
    page <- file.pages[page_id]

    // delete record
    page.records[rec_index].full = false
    page.number_of_records <- page.number_of_records - 1

    // update the next empty record index
    if page.next_empty_record_index = -1 then
        page.next_empty_record_index  <- rec_index
    elseif rec_index < page.next_empty_record_index
        page.next_empty_record_index <- rec_index
    end
end
```

4.2.3 Search for a record

```
function search_record(type_name,key_values[0:n])
    // find the index of type
    index <- 0
    while not (syscat.types[index].type_name = type_name) do
        index <- index + 1
    endwhile

    // get the file
    file_id <- syscat.types[index].file_id
```

```
    file <- getFile(file_id)

    // find the postion of each key corresponing to primary  key values
    primary_key_exists[0:9] <- syscat.types[index].primary_keys[0:9]
    key_index <- 0
    for i <- 1 to key_values.length do
        while primary_key_exists[key_index] = false do
            key_index <- key_index + 1
        endwhile
        key_pos[i-1] = key_index
    endfor

    // search through each page
    for i <- 0 to file.pages[].length do
        page <- file.pages[i]
        record_count <- page.number_of_records
        // if page is not empty
        if not ( record_count = 0) then
            for j <- 0 to record_count do
                // find the first non deleted record
                rec_index <- 0
                while page.records[rec_index].full = false do
                    rec_index <- rec_index + 1
                endwhile
                record <- page.records[rec_index]
                if record-match(record, key_pos, key_values, 0) = true then
                    file_id_s <- to_string(file_id)
                    page_id_s <- to_string(i)
                    rec_index_s <- to_string(rec_index)
                    return(str_concat(file_id_s, '/' , page_id_s , '/', rec_index_s))
                endif
            endfor
        endif
    endfor
    return('-1')
end

// returns true if the primary key values and the record field values match
function record-match(record,key_pos[0:n],key_values[0:n],index)
    if index >= key_values.length then
        return(true)
    else
       if record.fields[key_pos[index]] = key_values[index] then
           return(record-match(record,key_pos[0:n],key_values[0:n],index + 1))
       else
           return(false)
       endif
    endif
end
```

4.2.4 List all records of a type

```
function list_records(type_name)
    // find the index of type
    index <- 0
    while not (syscat.types[index].type_name = type_name) do
        index <- index + 1
    endwhile

    // get the file
    file_id <- syscat.types[index].file_id
    file <- getFile(file_id)

    // print all records if the page is not empty
    for i <- 0 to file.pages[].length - 1 do
        page <- file.pages[i]
        record_count <- page.number_of_records
        if not ( record_count = 0) then
            for j <- 0 to record_count - 1 do
                index <- 0
                while page.records[index].full = false do
                    index <- index + 1
                endwhile
                for k <- 0 to 9 do
                    print(page.records[index].field[k])
                endfor
            endfor
        endif
    endfor
end
```

5 Conclusions and Assessment

I created a very simple storage manager. It does the basic operations like adding, removing and listing types and records. Since the description said "do not include extra storage structures", I didn't implemented indexing. Therefore, my storage manager is slow and searches sequentially for records. To simplify the implementation, I used fixed length fields and stored each type in a different file.