Cmpe321:
Introduction to Database Systems
Homework 1

Spring 2019

Yağmur Ceren Dardağan
2014400063

Contents

1 Introduction

In this project, we are asked to design a storage manager system that stores data in storage units, pages and records, and implement DDL and DML operations.
DDL Operations:

- Create a type
- Delete a type
- List all types

DML Operations:

- Create a record
- Delete a record
- Search for a record (by primary key)
- List all records of a type

While implementing this project, I assumed user always enters valid input and fields are always integers and type names can be alphanumeric as the description stated.

2 Assumptions & Constraints

2.1 Constraints

- The data must be organized in pages and pages must contain records.
- A file must contain multiple pages and it is not allowed to put all the pages in one file.
- The system must be eligible to create new files as the manager grows.
- The system must be able to delete empty files due to deletions.
- The system must load a file page by page to RAM when it is needed.

2.2 Assumptions

- Page size is 1620 Byte(1,58 KB approximately) and every page has the same size.
- A page can store 15 records.
- Page header keeps the record of the page id, pointer to next page, location of next available record, number of non-empty records.
- Every record header is 7 byte(3 byte for record id, 3 byte for pointer to next record, 1 byte for isEmpty).
- Every page header is 15 bytes(3 bytes for page id, 4 bytes for pointer to next page, 4 bytes for pointer to next available record, 4 bytes for number of non-empty records).
- Record header keeps the record id,pointer of the next record, isEmpty information about the record.
- Every field is 10 byte and a record can store at most 10 fields.
- The length of a type name can be 20 characters at most(1 character is 1 byte)
- A file can contain 10 pages.
- File size is 15,8KB(approximately).
- Every file contains one type.

3 Data Structures

3.1 System Catalogue

System catalogue is responsible for storing the metadata that contains various tables and views and users are not allowed to modify these details. In this storage system, system catalogue consists of pages and every page contains a system catalogue page header and records.

- System Catalogue Page Header
  - Page ID
  - Pointer to Next Page

    - Pointer to next available record
    - Number of non-empty records

- Records
    - Record Header
        - Record ID
        - Pointer to next record
        - Number of fields
    - Data area

Record Id	Pointer to next Record	typeName	Number of Fields

Table 1: Record header for system catalogue

<table>
<tr><td colspan="5">Page Header</td></tr>
<tr><td>Record Header</td><td>Field Name 1</td><td>Field Name 2</td><td>...</td><td>Field Name 10</td></tr>
<tr><td>Record Header</td><td>Field Name 1</td><td>Field Name 2</td><td>...</td><td>Field Name 10</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>Record Header</td><td>Field Name 1</td><td>Field Name 2</td><td>...</td><td>Field Name 10</td></tr>
</table>

Table 2: a Page of a System Catalogue *(Starting with the Page Header)*

3.2 Page Design

A file contains more than one pages and a page contains a page header and records. A page contain only one type, so all the records stored in a page belongs to one type.

- Page Header (13 bytes)
    - Page ID
    - Pointer to Next Page

  - Pointer to Next Available Record
  - Number of non-empty records
- Records

Page Header
Record #1
Record #2
... ...
Record #15

Table 3: a Page with Records *(Starting with the Page Header)*

3.3 Record Design

A record contains a record header and data area to store actual data.

- Record Header (13 bytes)
  - Record ID
  - Pointer to Next Record
  - isEmpty information
- Actual Data

Record Id	Pointer to next Record	isEmpty

Table 4: Record header

Record Header
Field #1
Field #2
... ...
Field #10

Table 5: a Record with Fields *(Starting with the Record Header)*

4 Algorithms

4.1 DDL Operations

4.1.1 Create a type

Algorithm 1: Creating Data Type

```
createType(String typeName, String[] fieldNamesList, int numberOfFields)
create a data page with name typeName ;
pageAddress ← find firstPage of System Catalogue ;
while pageAddress is not NULL do
    read page via pageAddress(disk manager);
    if page→non_empty_records less then 10 then
        get record via page→next_available_record_pointer(disk manager);
        create new_record with typeName = typeName
        numberOfFields = numberOfFields and fields =
        fieldNamesList;
        insert new_record to record;
        page→non_empty_records ← page→non_empty_records + 1 ;
        return ;
    else
        pageAddress ← pageAddress→next ;
    end
end
create new_page with Page_id ← Page_id + 1 and page→non_empty_records ← 0 ;
pageAddress→next ← new_page ;
create new_record with typeName = typeName
numberOfFields = numberOfFields and fields =
fieldNamesList;
get record via new_page→next_available_record_pointer(disk manager);
insert new_record to record;
new_page→non_empty_records ← new_page→non_empty_records + 1 ;
```

4.1.2 Delete a type

Algorithm 2: Deleting Data Type

```
deleteType(String typeName)
delete data page with name typeName ;
pageAddress $\leftarrow$ find firstPage of System Catalogue ;
while pageAddress is not NULL do
    read page via pageAddress(disk manager);
    for all records in page do
        if record_typeName is typeName then
            make record_isEmpty = TRUE ;
              page$\rightarrow$non_empty_records $\leftarrow$ page$\rightarrow$non_empty_records -1 ;
            if page$\rightarrow$non_empty_records $\leq 0$ then
                delete page;
            end
            return ;
          else
            pageAddress $\leftarrow$ pageAddress$\rightarrow$next ;
          end
      end
  end
```

4.1.3 List all types

Algorithm 3: Deleting Data Type

```
listTypes()
pageAddress $\leftarrow$ find firstPage of System Catalogue ;
while pageAddress is not NULL do
    read page via pageAddress(disk manager);
    for all records in page do
        if record_isEmpty is FALSE then
            print record_typeName, record_fieldNamesList,
              record_numberOfFields;
        else
            pageAddress $\leftarrow$ pageAddress$\rightarrow$next ;
        end
    end
end
```

4.2 DML Operations

4.2.1 Create a record

Algorithm 4: Creating Record

```
createRecord(String typeName, String[] values_for_fields)
pageAddress ← find firstPage of data pages with name typeName;
while pageAddress is not NULL do
    read page via pageAddress(disk manager);
    if page→non_empty_records less then 10 then
        get record via page→next_available_record_pointer(disk
          manager);
        create new_record with typeName = typeName Fields =
          values_for_fields ;
        insert new_record to record;
        page→non_empty_records ← page→non_empty_records + 1 ;
        return ;
    else
        pageAddress ← pageAddress→next ;
    end
end
create new_page with Page_id ← Page_id + 1 and
  page→non_empty_records ← 0 ;
pageAddress→next ← new_page ;
create new_record with typeName = typeName Fields =
  values_for_fields ;
get record via new_page→next_available_record_pointer(disk
  manager);
insert new_record to record;
new_page→non_empty_records ← new_page→non_empty_records + 1 ;
```

4.2.2 Delete a record

Algorithm 5: Deleting Record

```
deleteRecord(String typeName, Int id)
delete data page with name typeName ;
pageAddress ← find firstPage of System Catalogue ;
while pageAddress is not NULL do
    read page via pageAddress(disk manager);
    for all records in page do
        if record_id equals id then
            make record_isEmpty = TRUE ;
              page→non_empty_records ← page→non_empty_records -1 ;
            if page→non_empty_records $\leq 0$ then
                delete page;
            end
            return ;
        else
            pageAddress ← pageAddress→next ;
        end
    end
end
```

4.2.3 Search for a record with primary key

Algorithm 6: Searching a Record with primary key

```
deleteRecord(String typeName, Int id)
delete data page with name typeName ;
pageAddress ← find firstPage of System Catalogue ;
while pageAddress is not NULL do
    read page via pageAddress(disk manager);
    for all records in page do
        if record_id equals id then
            return record_values ;
        else
            pageAddress ← pageAddress→next ;
        end
    end
end
```

4.2.4 List all records of a type

Algorithm 7: List all records for a type

```
deleteRecord(String typeName, Int id)
delete data page with name typeName ;
pageAddress ← find firstPage of System Catalogue ;
while pageAddress is not NULL do
    read page via pageAddress(disk manager);
    for all records in page do
        if record_isEmpty is FALSE then
            print record_values ;
        else
            pageAddress ← pageAddress→next ;
        end
    end
end
```

5 Conclusions & Assessment

In this assignment, I designed a storage manager system that includes system catalogue and data files structure for storing metadata and actual data. Some assumptions make my design process easier like assuming page and record header structures and sizes of the records,pages and files. However this design seems a little bit slow to search certain record in certain typename since it is searching pages one by one. In this design, the sizes of the each item in record,pages and files can be listed as follows;

Record design:

- Record Header(7 bytes)
    - Record Id(3 bytes)
    - Pointer to next record(3 bytes)
    - isEmpty(1 bytes)
- Data area(10 bytes for each field, max. 10 fields)

Page design:

- Page Header(15 bytes)
    - Page Id(3 bytes)

    - Pointer to next page(4 bytes)
    - Pointer to next available record(4 bytes)
    - Number of non-empty records(4 bytes)

- Records(max 107 bytes for a record, max 15 records)

File design:

- Pages(max 1620 bytes for a page, max 10 pages)

The design of this storage manager system seems logical to me but I cannot be sure before implementing the system and I will change the inconsistencies during implementation in the second assignment.