
CMPE321:
INTRODUCTION TO DATABASE SYSTEMS
HOMEWORK 2

SPRING 2019

YAĞMUR CEREN DARDAĞAN
2014400063

Contents

1	Introduction	2
2	Assumptions & Constraints	2
2.1	Constraints	2
2.2	Assumptions	3
2.2.1	Assumptions in Homework 1	3
2.2.2	Assumptions in Homework 2	3
3	Data Structures	4
3.1	System Catalogue	4
3.2	Page Design	5
3.3	Record Design	6
4	Algorithms	6
4.1	DDL Operations	7
4.1.1	Create a type	7
4.1.2	Delete a type	8
4.1.3	List all types	8
4.2	DML Operations	9
4.2.1	Create a record	9
4.2.2	Delete a record	10
4.2.3	Search for a record with primary key	10
4.2.4	List all records of a type	11
5	Conclusions & Assessment	11

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

2.2.1 Assumptions in Homework 1

- 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.

2.2.2 Assumptions in Homework 2

- Page size is 420 Byte(0,410 KB approximately) and every page has the same size.
- A page can store 10 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 4 byte(4 byte for isEmpty).

- Every page header is 20 bytes(4 bytes for page id, 6 bytes for pointer to next page, 6 bytes for pointer to next available record, 4 bytes for number of non-empty records).
- Record header keeps the record id,isEmpty information about the record.
- Every field is 10 byte and a record can store at most 6 fields.
- The length of a type name can be 10 characters at most(1 character is 1 byte)
- A file can contain 10 pages.
- File size is 4,10KB(approximately).
- Every file contains one type.
- In system catalogue, every page header keeps the page id and number of records information(4 byte for page id, 4 bytes for number of record information).
- In system catalogue, every record consists of type id(4 bytes), type name(10 bytes), number of files(4 bytes), number of pages(4 bytes), number of records(4 bytes), number of fields(4 bytes) and fields names(10 bytes each, max 6 fields.)

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. The information for the types has changed when it is compared with homework one. Because in implementation, the values like number of pages or records are important for keeping track of the records of a specific type.

- System Catalogue Page Header
 - Page ID
 - Total number of records in page

- Records
 - Record Header
 - * type Id(4 bytes)
 - * type Name(10 bytes)
 - * number Of Files(4 bytes)
 - * number Of Pages(4 bytes)
 - * number Of Records(4 bytes)
 - * number Of Fields(4 bytes)
 - Fields(10 bytes each, max 6 fields)

type Id	type name	number of files	number of pages	number of records	Number of Field
---------	-----------	-----------------	-----------------	-------------------	-----------------

Table 1: Record header for system catalogue

Page Header				
Record Header	Field Name 1	Field Name 2	...	Field Name 10
Record Header	Field Name 1	Field Name 2	...	Field Name 10
...
Record Header	Field Name 1	Field Name 2	...	Field Name 10

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. It remained same as the homework one, the design was enough to implement.

- Page Header (20 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 id and pointer to next record information have been removed, because it was complicated and unnecessary to keep that information for the sake of this implementation.

- Record Header (4 bytes)
 - isEmpty information
- Actual Data

isEmpty

Table 4: Record header

4 Algorithms

The algorithms that has been planned was implemented in the homework 2, there has been no change in algorithm.

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

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

4.1 DDL Operations

4.1.1 Create a type

Algorithm 1: Creating Data Type

```

1 createType(String typeName, String[] fieldNamesList, int
  numberOfFields)
2 create a data page with name typeName ;
3 pageAddress  $\leftarrow$  find firstPage of System Catalogue ;
4 while pageAddress is not NULL do
5   read page via pageAddress(disk manager);
6   if page $\rightarrow$ non_empty_records less than 10 then
7     get record via page $\rightarrow$ next_available_record_pointer(disk
      manager);
8     create new_record with typeName = typeName
9     numberOfFields = numberOfFields and fields =
10    fieldNamesList;
11    insert new_record to record;
12    page $\rightarrow$ non_empty_records  $\leftarrow$  page $\rightarrow$ non_empty_records + 1 ;
13    return ;
14  else
15    pageAddress  $\leftarrow$  pageAddress $\rightarrow$ next ;
16  end
17 end
18 create new_page with Page_id  $\leftarrow$  Page_id + 1 and
  page $\rightarrow$ non_empty_records  $\leftarrow$  0 ;
19 pageAddress $\rightarrow$ next  $\leftarrow$  new_page ;
20 create new_record with typeName = typeName
21 numberOfFields = numberOfFields and fields =
22 fieldNamesList;
23 get record via new_page $\rightarrow$ next_available_record_pointer(disk
  manager);
24 insert new_record to record;
25 new_page $\rightarrow$ non_empty_records  $\leftarrow$  new_page $\rightarrow$ non_empty_records + 1 ;

```


4.1.2 Delete a type

Algorithm 2: Deleting Data Type

```
1 deleteType(String typeName)
2 delete data page with name typeName ;
3 pageAddress  $\leftarrow$  find firstPage of System Catalogue ;
4 while pageAddress is not NULL do
5   read page via pageAddress(disk manager);
6   for all records in page do
7     if record_typeName is typeName then
8       make record_isEmpty = TRUE ;
9       page $\rightarrow$ non_empty_records  $\leftarrow$  page $\rightarrow$ non_empty_records -1 ;
10      if page $\rightarrow$ non_empty_records  $\leq$  0 then
11        delete page;
12      end
13      return ;
14    else
15      | pageAddress  $\leftarrow$  pageAddress $\rightarrow$ next ;
16    end
17 end
```

4.1.3 List all types

Algorithm 3: Deleting Data Type

```
1 listTypes()
2 pageAddress  $\leftarrow$  find firstPage of System Catalogue ;
3 while pageAddress is not NULL do
4   read page via pageAddress(disk manager);
5   for all records in page do
6     if record_isEmpty is FALSE then
7       print record_typeName, record_fieldNamesList,
8         record_numberOfFields;
9     else
10      | pageAddress  $\leftarrow$  pageAddress $\rightarrow$ next ;
11    end
12 end
```

4.2 DML Operations

4.2.1 Create a record

Algorithm 4: Creating Record

```
1 createRecord(String typeName, String[] values_for_fields)
2 pageAddress ← find firstPage of data pages with name typeName;
3 while pageAddress is not NULL do
4   read page via pageAddress(disk manager);
5   if page→non_empty_records less than 10 then
6     get record via page→next_available_record_pointer(disk
7       manager);
8     create new_record with typeName = typeName Fields =
9       values_for_fields ;
10    insert new_record to record;
11    page→non_empty_records ← page→non_empty_records + 1 ;
12    return ;
13  else
14    pageAddress ← pageAddress→next ;
15  end
16 end
17 create new_page with Page_id ← Page_id + 1 and
18   page→non_empty_records ← 0 ;
19 pageAddress→next ← new_page ;
20 create new_record with typeName = typeName Fields =
21   values_for_fields ;
22 get record via new_page→next_available_record_pointer(disk
23   manager);
24 insert new_record to record;
25 new_page→non_empty_records ← new_page→non_empty_records + 1 ;
```

4.2.2 Delete a record

Algorithm 5: Deleting Record

```
1 deleteRecord(String typeName, Int id)
2 delete data page with name typeName ;
3 pageAddress  $\leftarrow$  find firstPage of System Catalogue ;
4 while pageAddress is not NULL do
5   read page via pageAddress(disk manager);
6   for all records in page do
7     if record_id equals id then
8       make record_isEmpty = TRUE ;
9       page $\rightarrow$ non_empty_records  $\leftarrow$  page $\rightarrow$ non_empty_records -1 ;
10      if page $\rightarrow$ non_empty_records  $\leq$  0 then
11        | delete page;
12      end
13      return ;
14    else
15      | pageAddress  $\leftarrow$  pageAddress $\rightarrow$ next ;
16    end
17  end
```

4.2.3 Search for a record with primary key

Algorithm 6: Searching a Record with primary key

```
1 deleteRecord(String typeName, Int id)
2 delete data page with name typeName ;
3 pageAddress  $\leftarrow$  find firstPage of System Catalogue ;
4 while pageAddress is not NULL do
5   read page via pageAddress(disk manager);
6   for all records in page do
7     if record_id equals id then
8       | return record_values ;
9     else
10      | pageAddress  $\leftarrow$  pageAddress $\rightarrow$ next ;
11    end
12  end
13 end
```

4.2.4 List all records of a type

Algorithm 7: List all records for a type

```
1 deleteRecord(String typeName, Int id)
2 delete data page with name typeName ;
3 pageAddress ← find firstPage of System Catalogue ;
4 while pageAddress is not NULL do
5   | read page via pageAddress(disk manager);
6   | for all records in page do
7     | if record.isEmpty is FALSE then
8       |   | print record_values ;
9       |   else
10      |     | pageAddress ← pageAddress→next ;
11      |     end
12    end
13 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(4 bytes)
 - isEmpty(4 bytes)
- Data area(6 bytes for each field, max. 6 fields)

Page design:

- Page Header(20 bytes)
 - Page Id(4 bytes)
 - Number of non-empty records(4 bytes)

- Pointer to next available record(6 bytes)
- Pointer to next page(6 bytes)
- Records(max 40 bytes for a record, max 10 records)

File design:

- Pages(max 420 bytes for a page, max 10 pages)

In System Catalogue:

- Page Header(8 bytes)
 - Number of types(4 bytes)
 - Number of pages(4 bytes)
- Records(max 90 bytes for a record)
 - type Id(4 bytes)
 - type Name(10 bytes)
 - number Of Files(4 bytes)
 - number Of Pages(4 bytes)
 - number Of Records(4 bytes)
 - number Of Fields(4 bytes)
 - Fields(10 bytes each, max 6 fields)

In the implementation homework, homework 2, most part of the homework 1 stayed same but the sizes of page header,record header, records,pages and gradually files have been changed. I changed the contents of record header and page header for the sake of implementation. It is working generally, but I doubt about the types that have number of records that more than 1 file.It is working for 100 records correctly.