Storage System Manager Design

Ahmet Gedemenli - 2014400147

July 2019

Project 2, CMPE321 Introduction to Database Systems, 2019 Summer

1 Introduction

This LaTeXdocument is a report that describes a storage manager implementation. It allows you to do some basic operations.

Allowed operations are:

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

Java is used for implementing the storage manager. The assumptions, constraints and the structures can be found at the next pages. Operations are written in a very basic, limited format in this report. They can be found more specific in Java code of classes and methods which can be found in /src. The changes from the first project can be found also in this report. The report ends with the author's evaluations and comments in the conclusion page.

Please note that this implementation does not include any error handling, means getting a valid input is assumed. You can find further details especially in the Assumptions page.

2 Assumptions & Constraints

Below assumptions and constraints about the project can be found.

- User always enters valid input.
- The codes in the operations part are really dummy and useless. The real Java code can be found and tested in the folder src.
- All fields shall be integers. However, type and field names shall be alphanumeric.
- All fields(no matter in records or page headers etc.) can have a name with maximum 10 characters. This limit was 20 in the design of first project. However it is not 10 due to the limits of this second project.
- One page can contain just one type of record.
- Creating records of the same type with the same primary key is not allowed.
- Creating a record of a type which does not exist is not allowed.
- Updating a record which does not exist is not allowed.
- All sort of invalid input is not allowed. The implementation does not handle these cases.
- There is no primary key seperated from the fields. Primary key is always the first field of a record; different from the design of the first project.

3 Storage Structures

The System Catalog contains records(all types) with features such as # of fields, # of header fields.The metadata of files and pages are stored here.

The data types are,

- File

Page #1	...

- Page

Page Header	Record1	Record2	...
10 Byte	...	...	...

- Page Header

pageId	int # of records	pointer to the next page	isFull	isEmpty
4 Byte	4 Byte	14bit	1bit	1bit

- Record

Record Header	int field1(primary key)	int field2	...
8 Byte	4 Byte	4 Byte	...

- Record Header

int recordType	int # of records
4 Byte	4 Byte

4 DDL Operations

These dummy and useless methods are written just to give a point of view, in the first project. The real Java code can be found and tested in the folder named src.

4.1 Create a type

```
// Let's say we have an array named Types.
// Parameters are the inputs given by the user. Respectively String, int, int.

function createType(typeName, recordType, cntOfFields) begin

1— newType.typeName = typeName
2— newType.recordType = recordType
3— newType.cntOfFields = cntOfFields
4— Types.add(newType);
end
```

4.2 Delete a type

```
// Parameter is the input(int) given by the user.
// It's the id of the record to be deleted.

function deleteType(recordType) begin

1— Types.remove(recordType)
end
```

4.3 List all types

```
// We just need to print all of the array named Types. No parameter.

function listAllTypes()

1— foreach type in Types do
2 — print(type)
3— endfor
end
```

5 DML Operations

Let's say we have the current page in a variable named page.

5.1 Create a record

```
// Here parameters are both int.

function createRecord(recordType, key) begin
1— newRecord.recordType = recordType
2— newRecord.id = key
3— cnt = 0
4— while(cnt <recordType.numOfFields) do
5 —     newRecord.addField()
6 —     cnt++
7— end
8— page.add(recordType,newRecord);
end
```

5.2 Delete a record

```
// The parameter is int.

function deleteRecord(key) begin
1— page.removeById(key)
end
```

5.3 Search a record

```
// The parameter is int.

function searchRecord(key) begin
1— return page.getElementById(key)
end
```

5.4 Update a record

```
// The parameters are both int.

function updateRecord(recordType, key) begin
1— deleteRecord(key)
2— createRecord(recordType, key)
end
```

5.5 List all record of a type

```
// The parameter is int.

function listAllRecordsOfAType(recordType) begin
1— foreach record in page begin
2 — print(record)
3— endfor
end
```

6 Conclusions & Assessment

In this report I tried to describe the differences on my design from the first project. This report mainly emphasizes the changes on the design. The Java implementation of this system can be found in the folder named src.

Working on this design made me understand how databases work in the very basic level. This one can evolve to a much more complex and powerful system but it's very simple now. I hope the next project will also improve my skills and perceptions about databases.