# CMPE 321 Assignment 1 Report

**Course Name:** Introduction to Database Systems

**Semester:** Spring, 2017

**Assignment Title:** Data Storage Manager Design

**Student Name:** Rahmetullah Varol

# 1 Introduction

The aim of this project is to design and implement a simple database storage management system. In this report, the design details of this storage manager are presented. The assumptions and constraints for the design are specified. In the determination of these parameters, performance optimisation was prioritised while trying to keep the size requirements at a reasonable level.

Using the specified constraints and assumptions as a guide, data structures used in the design of the storage manager are specified. These data structures give details about the structure of the page and record headers inside catalogue and data files. They help organisation of data inside these files and make access and manipulation tasks on records easier.

Next, pseudocodes of the algorithms for operations which a user can perform on the database are given. In the design of these algorithms, the specified data structures are taken into account. With such a unified approach, implementation of this storage manager system is just a matter of translation to the preferred programming language.

# 2 Assumptions and Constraints

The database is designed with the following constraints:

Page size: 2 kB
Maximum number of fields: 16
Maximum length of type name: 24 characters
Maximum length of field name: 24 characters
No two types may have the same name

All field types are assumed to be integers. The field names are assumed to be alphanumeric.

# 3 Data Structures

There are two different types files in this project: Catalogue files and data files. Catalogue files are where the information about different types of data

Table 1: Structure of catalogue and data files.

| | | | | | | |
|---|---|---|---|---|---|---|
| Page Header 1 | Record Header 1 | Field 1 | ... | RH 2 | Field 1 | ... |
| Page Header 2 | Record Header 1 | Field 1 | ... | RH 2 | Field 1 | ... |
| ... | ... | ... | ... | ... | ... | ... |

are stored. This information consists of the type name and the field names for each field. On the otherhand, data files contain the values for each of the fields. Both catalogue files and data files are structured as in Table 1.

The information to be stored in the page header and the record header of data files are as follows:

| **Page header:** | **Record header:** |
|---|---|
| pageID | recordID |
| remainingSize | fieldCount |
| numberOfRecords | isEmpty |
| isLast | |

In catalogue files' record headers, in addition to the stated fields there also exists a field named 'typeName' in which the name of the type that the record is associated with is kept.

# 4 Operations

In this section, algorithms for certain operations a user can perform are given.

The first algorithm is for the operation through which a user can create a new type.

**Algorithm 1** Algorithm for creating a type

```
procedure CreateType
    Open System Catalogue file
    $typeName \leftarrow$ Type name from user
    $fieldCount \leftarrow$ Number of fields from user
    for $i$ from 0 to $fieldCount$ do
        $fieldNames[i] \leftarrow$ Field name from user
    end for
    for each $page$ in System Catalogue do
        if page.isLast then
            if $page.recordCount < maxRecordCount$ then
                $newRecord \leftarrow$ Create a new record
                $newRecord.typeName \leftarrow typeName$
                $newRecord.fieldCount \leftarrow fieldCount$
                $newRecord.isEmpty \leftarrow$ False
                for $i$ from 0 to $fieldCount$ do
                    $newRecord.fieldName[i] \leftarrow fieldNames[i]$
                end for
                Add $newRecord$ to $1page$
                $page.remainingSize \leftarrow page.remainingSize - recordSize$
                $page.numberOfRecords \leftarrow page.numberOfRecords + 1$
            else
                $newPage \leftarrow$ Create a new page
                $newPage.remainingSize$ -= $pageHeaderSize$
                $newPage.isLast \leftarrow$ True
                $page.isLast \leftarrow$ False
            end if
        end if
    end for
    Create a new file named '$typeName$'.dat
end procedure
```

The second algorithm is for the operation through which a user can delete an existing type using it's primary key.

**Algorithm 2** Algorithm for deleting a type

```
procedure DeleteType
    Open System Catalogue file
    $typeID \leftarrow$ Type ID from user
    for each $page$ in System Catalogue do
        $found \leftarrow$ False
        for each $record$ in $page$ do
            if $record.id == typeID$ then
                $record.isEmpty \leftarrow$ True
                $page.remainingSize \leftarrow page.remainingSize + recordSize$
                $page.numberOfFields \leftarrow page.numberOfFields - 1$
                $found \leftarrow$ True
                break
            end if
            if $found$ then
                break
            end if
        end for
    end for
    Delete the file named '$typeName$'.dat
end procedure
```

The third algorithm is for the operation through which a user can list all the types that currently exist in the database.

**Algorithm 3** Algorithm for listing all types in a database

```
procedure ListTypes
    Open the System Catalogue file
    for each page in System Catalogue do
        for each record in page do
            if not record.isEmpty then
                Print record.typeName
                for each field in record do
                    Print field.name
                end for
            end if
        end for
    end for
end procedure
```

The fourth algorithm is for the operation through which a user can create a new record for a given type.

**Algorithm 4** Algorithm for creating a record for a given type

```
procedure CreateRecord
    typeName ← Type name from user
    Open System Catalogue file
    for each page in System Catalogue do
        for each record in page do
            if record.typeName  == typeName then
                fieldCount ← record.fieldCount
            end if
        end for
    end for
    Open the file named 'typeName'.dat
    for each page in Data File do
        if page.isLast then
            if page.recordCount < maxRecordCount then
                newRecord ← Create a new record
                newRecord.typeName ← typeName
                newRecord.fieldCount ← fieldCount
                newRecord.isEmpty ← False
                for i from 0 to fieldCount do
                    newRecord.fieldValue[i] ← Get field value from user
                end for
                Add newRecord to 1page
                $page.remainingSize \leftarrow page.remainingSize - recordSize$
                $page.numberOfRecords \leftarrow page.numberOfRecords + 1$
            else
                newPage ← Create a new page
                newPage.remainingSize -= pageHeaderSize
                newPage.isLast ← True
                page.isLast ← False
            end if
        end if
    end for
end procedure
```

The fifth algorithm is for the operation through which a user can delete an existing record for a given type.

**Algorithm 5** Algorithm for deleting a record

```
procedure DeleteRecord
    $typeName \leftarrow$ Type name from user
    $recordID \leftarrow$ Record ID from user
    Open the file named '$typeName$'.dat
    for each $page$ in Data File do
        $found \leftarrow$ False
        for each $record$ in $page$ do
            if $record.id == typeID$ then
                $record.isEmpty \leftarrow$ True
                $page.remainingSize \leftarrow page.remainingSize + recordSize$
                $page.numberOfFields \leftarrow page.numberOfFields - 1$
                $found \leftarrow$ True
                break
            end if
            if $found$ then
                break
            end if
        end for
    end for
end procedure
```

The sixth algorithm is for the operation through which a user can update certain fields of an existing record.

**Algorithm 6** Algorithm for updating a record

```
procedure UpdateRecord
    recordID ← Record ID from user
    Open System Catalogue file
    for each page in System Catalogue do
        for each record in page do
            if record.typeName  == typeName then
                typeName ← record.typeName
                fieldCount ← record.fieldCount
            end if
        end for
    end for
    Open the file named 'typeName'.dat
    for each page in Data File do
        for each record in page do
            if record.id == recordID then
                for i from 0 to fieldCount do
                    record.fieldValue[i] ← Get new field value from user
                end for
            end if
        end for
    end for
end procedure
```

The seventh algorithm is for the operation through which a user can search for an existing record using it's primary key.

**Algorithm 7** Algorithm for searching for a record

```
procedure SearchRecord
    typeName $\leftarrow$ Type name from user
    recordID $\leftarrow$ Record ID from user
    Open the file named 'typeName'.dat
    for each page in Data File do
        found $\leftarrow$ False
        for each record in page do
            if record.id == typeID then
                Return record
            end if
        end for
    end for
end procedure
```

The eighth algorithm is for the operation through which a user can list all the current records of a given type.

**Algorithm 8** Algorithm for all records of a type

```
procedure ListRecords
    $typeName \leftarrow$ Type name from user
    Open System Catalogue file
    for each $page$ in System Catalogue do
        for each $record$ in $page$ do
            if $record.typeName == typeName$ then
                $i \leftarrow 0$
                for each $field$ in $record$ do
                    $fieldNames[i] \leftarrow field.name$
                    $i \leftarrow i + 1$
                end for
            end if
        end for
    end for
    Open the file named '$typeName$'.dat
    for each $page$ in Data File do
        for each $record$ in $page$ do
            Print $record.id$
            $i \leftarrow 0$
            for each $field$ in $record$ do
                Print $fieldNames[i] +' :' + field$
                $i \leftarrow i + 1$
            end for
        end for
    end for
end procedure
```

# 5 Conclusions and Assessments

In this document, the design parameters of a simple storage manager has been clearly laid out. These parameters will be used in the implementation phase of this project. One weakness of this design is that multiple types with the same name are not allowed. Since the manager differentiates between different types using their name and not some other unique entity, using two different types with the same name will cause the system to confuse these types.

Keeping isLast flags in pages and isEmpty flags in records allow quick creation of new records. These are good examples of how the headers can be utilized to increase the performance of the storage manager. Similar improvements can be made to further increase the performance.