Bogazici University

cmpe 321

Introduction to Database Systems

Design of Storage Manager

Author: Ahmet Ege MAHLEC
2016800045

March 26, 2017

1 Introduction

This report is dedicated to explain the design of Storage Manager which is one module of database management system. The program, whose design is explained in this report, allows the Data Definition Language (DDL) and Data Manipulation Language (DML) operations which are given below.

DDL Operations

1. Create a type
2. Delete a type
3. List all types

DML Operations

1. Create a record
2. Delete a record
3. Update a record
4. Search for a record via primary key
5. List all records of a type

The storage manager creates two types of files, namely, system catalog and data files. When user requests to create a new type, the meta data of this type will be stored in the system catalog. In contrast, when user requests to delete a type, the meta data of this type will be removed from system catalog.

The other file type which was created by storage manager is data files. When user requests to create a new type, data file of this type will be created unless it has been created. When user requests to delete a record, data file which belongs to this type will be removed. Also, when user request to add a new record, this new record will be added to data file of this type.

2 Assumptions and Constraints

In this section, the design choices will be explained by stating the assumptions and constraints which are used to implement Storage Manager.

- Characters which is not defined in ASCII table will not be given as input.
- Storage manager does not give any service to interact tables with each other.
- Files shall be divided in the pages.
- In order to read or write data, whole page shall be read or written.
- Page size shall be 1 KB.
- Length of the type name shall be at most 16 bytes long.
- Length of the field name shall be at most 16 bytes long.
- Type and field name shall be alphanumeric.
- Every type has at most 10 fields.
- The value of all fields will be type signed integer.
- The value of all fields will be at most 4-byte long.
- User is not allowed to update a type once it has been created. It should be deleted and created again if a record would like to be updated. In this case, all the records will be deleted.
- All the data will be written in little endian format.
- First field of the record will be selected as primary key.
- User will not enter any record whose primary key is same with the record which has already exist in the database.
- User will not enter a type whose name is same with type which has already exist in the database.
- User will not try to delete any record or type which doesn't exist in the database.

3 Data Structures

3.1 System Catalog File Format

System catalog is composed of the pages. Every page has header part and data part.

Table 1: System Catalog File Format

Page Header	T#1 Header	T#1 F#1	T#1 F#2	...	T#1 F#10
T#2 Header	T#2 F#1	T#2 F#2	T#2 F#3	...	T#2 F#10
T#5 Header	T#5 F#1	T#5 F#2	T#5 F#3	...	T#5 F#10

Table 2: Page Header Format

PageID (4 byte)	TotalSpace(2 byte)	FreeSpace(2 byte)	Reserved(116 byte)

Table 3: Type Header Format

TypeID (1 byte)	isUsed(1 byte)	NumberOfField(1 byte)	TypeName(4bytes)	Reserved(13 byte)

As it is shown in Table 1, System Catalog File starts with a Page Header and follows by a type header and field name of the types. Since every type has at most 10 fields; if a type is created, 10 x 16 + 20= 180 bytes allocated for this type even if it has less fields. After page is full, other page is generated. The other page will have the same format with the first format. It will have page header followed by type header and field name of the types.

As it is shown in Table 2, page header has ”PageID”, ”TotalSpace” and ”FreeSpace” information. 116 bytes are reserved for future use. It is important to note that PageID describes how far away from the beginning of the file. If all the types is deleted in a page, page will still be allocated and its FreeSpace will be equal to TotalSpace. In other words, once the page is created, it will not be deleted any more.

As it is shown in Table 3, type header has ”TypeID”, ”isUsed” and ”NumberOfField” information. 17 bytes are reserved for future use. TypeID describes how far away from the beginning of the page. Even if the type is deleted, the space which is allocated for this type will not be deallocated. The program will just reset ”isUsed” flag.

3.2 Data File Format

Data Files are also composed of the pages. Every page has header part and data part.

Table 4: Data File Format

Page Header	R#1 Header	R#1 F#1	R#1 F#2	...	R#1 F#10
R#2 Header	R#2 F#1	R#2 F#2	R#2 F#3	...	R#2 F#10
R#5 Header	R#5 F#1	R#5 F#2	R#5 F#3	...	R#5 F#10

Table 5: Page Header Format

PageID (4 byte)	TotalSpace(2 byte)	FreeSpace(2 byte)	Reserved(116 byte)

Table 6: Record Header Format

RecordID (1 byte)	isUsed(1 byte)	NumberOfField(1 byte)	Reserved(17 byte)

As it is shown in Table 4, Data Files start with a Page Header and follows by a record header and values of the its fields. Since every record has at most 10 fields; if a record is created, 10 x 16 + 20= 180 bytes allocated for this type even if it has less fields. After page is full, other page is generated. The other page will have the same format with the first format. It will have page header followed by record header and values of its fields.

As it is shown in Table 5, page header has ”PageID”, ”TotalSpace” and ”FreeSpace” information. 116 bytes are reserved for future use. It is important to note that PageID describes how far away from the beginning of the file. If all the records of this page is deleted in a page, page will still be allocated and its FreeSpace will be equal to TotalSpace. In other words, once the page is created, it will not be deleted any more.

As it is shown in Table 6, record header has "RecordID", "isUsed" and "NumberOfField" information. 17 bytes are reserved for future use. RecordID describes how far away from the beginning of the page. Even if the record is deleted, the space which is allocated for this type will not be deallocated. The program will just reset "isUsed" flag.

4 Algorithms

In this section, pseudo code of DML and DDL algorithms will be presented.

4.1 DDL Operations

Algorithm 1 Create a Type

```
procedure CreateType($TypeName$, $\#OfField$, $FieldNames$)
    if $TypeName.datfile\ does\ not\ exist$ then
        $Create\ TypeName.datfile$
    end if
    if $Open\ sys.cat\ file\ is\ false$ then
        $Create\ sys.catfile$
        $Open\ sys.cat\ file$
    end if
    $bPageFound \leftarrow false, nPageID \leftarrow 0$
    while $bPageFound\ is\ false$ do
        $nAddress \leftarrow nPageID * 1024$
        if $FileSize\ is\ greater\ or\ equal\ to\ (nPageID + 1) * 1024$ then
            $Read\ whole\ page\ from\ sys.cat\ file\ beginning\ from\ nAddress$
        else
            $Allocate\ 1024\ Bytes\ for\ new\ page$
            $Create\ its\ page\ header$
        end if
        $Read\ FreeSpace\ from\ Page\ Header$
        if $FreeSpace\ of\ the\ page\ is\ greater\ or\ equal\ to\ 180KB$ then
            $nTypeID \leftarrow 0, bWritten \leftarrow false$
            while $bWritten\ is\ false$ do
                if $isUsed\ flag\ of\ Type\ whose\ ID\ is\ nTypeID\ is\ false$ then
                    $nFieldID \leftarrow 0$
                    while $nFieldID < \#OfField$ do
                        $Write\ FieldNames\ 16\ bytes\ for\ each$
                        $nFieldID \leftarrow nFieldID + 1$
                    end while
                    $Update\ Record\ and\ Type\ Header$
                    $bWritten \leftarrow true$
                    $bPageFound \leftarrow true$
                    $Write\ whole\ page\ data\ from\ buffer\ to\ sys.cat\ file$
                end if
                $nTypeID \leftarrow nTypeID + 1$
            end while
        end if
        $nPageID \leftarrow nPageID + 1$
    end while
end procedure
```

Algorithm 2 Delete a Type

```
procedure DeleteType(TypeName)
    if TypeName.datfile exists then
        Delete TypeName.datfile
    end if
    if Open sys.cat file is false then
        return false
    end if
    $TypeFound \leftarrow false, nPageID \leftarrow 0$
    while TypeFound is false do
        $nAddress \leftarrow nPageID * 1024$
        if FileSize is greater or equal to $(nPageID + 1) * 1024$ then
            Read whole page from sys.cat file beginning from nAddress
        end if
        Read FreeSpace and TotalSpace of Page
        $nTypeID \leftarrow 0$
        while $((nTypeID+1)*180)$ is less or equal to $(TotalSpace-124)$ do
            if isUsed of Type whose ID is nTypeID is true then
                Get TypeName of the type whose ID is nTypeID
                if TypeName of Type is same with input TypeName then
                    isUsed flag of Type whose ID is nTypeID $\leftarrow 0$
                    Update Record and Page Header
                    Write whole page from buffer to sys.cat
                    $bTypeFound \leftarrow true$
                    return true
                end if
            end if
            $nTypeID \leftarrow nTypeID + 1$
        end while
        $nPageID \leftarrow nPageID + 1$
    end while
    if bTypeFound is false then
        return false
    end if
end procedure
```

Algorithm 3 List All Types

```
procedure ListAllTypes
    $Define\ a\ list\ to\ store\ TypeNames\ and\ their\ fields$
    if $Open\ sys.cat\ file\ is\ false$ then
        $returnemptylist$
    end if
    $nPageID \leftarrow 0$
    while $(nPageID+1)*1024$ $is\ less\ than\ or\ equal\ to\ sys.cat\ file\ size$ do
        $Read\ whole\ page\ whose\ id\ is\ nPageID$
        $nTypeID \leftarrow 0$
        while $(nTypeID + 1) * 180$ $is\ less\ than\ or\ equal\ to$ $(1024 - 124)$ do
            if $isUsed\ flag\ of\ type\ whose\ ID\ is\ nTypeID\ is\ true$ then
                $Add\ TypeName\ and\ its\ fields\ to\ the\ list$
            end if
            $nTypeID \leftarrow nTypeID + 1$
        end while
        $nPageID \leftarrow nPageID + 1$
    end while
    $Return\ the\ list$
end procedure
```

4.2 DML Operations

Algorithm 4 Create A Record

```
procedure CreateRecord($TypeName$,$Record$)
    $Open\ TypeName.dat\ file$
    $nPageID \leftarrow 0, bRecordCreated \leftarrow false$
    while ($(nPageID + 1) * 1024\ is\ less\ than\ or\ equal\ to\ file\ size$ do
        $Read\ whole\ page\ whose\ ID\ is\ nPageID$
        if $FreeSpace\ of\ page\ is\ less\ or\ equal\ to\ 180KB$ then
            $nRecordID \leftarrow 0$
            while $(nRecordID + 1) * 180\ is\ less\ than\ or\ equal\ to\ (1024 - 124)$ do
                if $isUsed\ flag\ of\ Record\ is\ 0$ then
                    $Write\ Record\ Information$
                    $Update\ Record\ and\ Page\ Header$
                    $Write\ whole\ page\ from\ buffer\ to\ file$
                    $bRecordCreated \leftarrow true$
                    $Return\ true$
                end if
                $nRecordID \leftarrow nRecordID + 1$
            end while
        end if
        $nPageID \leftarrow nPageID + 1$
    end while
    if $bRecordCreated\ is\ false$ then
        $Create\ a\ new\ page\ with\ page\ header$
        $Write\ Record\ Information$
        $Update\ Record\ and\ Page\ Header$
        $Write\ whole\ page\ from\ buffer\ to\ file$
        $bRecordCreated \leftarrow true$
        $Return\ true$
    end if
end procedure
```

Algorithm 5 Delete A Record

```
procedure DeleteRecord($TypeName$,$Record$)
    $Open\ TypeName.dat\ file$
    $nPageID \leftarrow 0, bRecordDeleted \leftarrow false$
    while $((nPageID + 1) * 1024$ $is\ less\ than\ or\ equal\ to\ file\ size$ do
        $Read\ whole\ page\ whose\ ID\ is\ nPageID$
        if $(TotalSpace - FreeSpace)$ $of\ page\ is\ greater\ or\ equal\ to$ $(180 + 124)$ $Bytes$ then
            $nRecordID \leftarrow 0$
            while $(nRecordID + 1) * 180$ $is\ less\ than\ or\ equal\ to$ $(1024 - 124)$ do
                if $isUsed\ flag\ of\ Record\ is$ 1 then
                    if $Record\ primary\ key\ is\ same\ with\ input\ Record\ primary\ key$ then
                        $isUsed\ flag\ of\ Record \leftarrow 0$
                        $Update\ Record\ and\ Page\ Header$
                        $Write\ whole\ page\ from\ buffer\ to\ file$
                        $bRecordDeleted \leftarrow true$
                        $return\ true$
                    end if
                end if
                $nRecordID \leftarrow nRecordID + 1$
            end while
        end if
        $nPageID \leftarrow nPageID + 1$
    end while
    if $bRecordDeleted\ is\ false$ then
        $Return\ false$
    end if
end procedure
```

Algorithm 6 Update A Record

```
procedure UpdateRecord($TypeName$,$Record$)
    $Open\ TypeName.dat\ file$
    $nPageID \leftarrow 0, bRecordUpdated \leftarrow false$
    while $((nPageID + 1) * 1024$ $is\ less\ than\ or\ equal\ to\ file\ size$ do
        $Read\ whole\ page\ whose\ ID\ is\ nPageID$
        if $(TotalSpace - FreeSpace)$ $of\ page\ is\ greater\ or\ equal\ to$ $(180 + 124)$ $bytes$ then
            $nRecordID \leftarrow 0$
            while $(nRecordID + 1) * 180$ $is\ less\ than\ or\ equal\ to$ $(1024 - 124)$ do
                if $isUsed\ flag\ of\ Record\ is\ 1$ then
                    if $Record\ primary\ key\ is\ same\ with\ input\ Record\ primary\ key$ then
                        $Overwrite\ input\ record\ info$
                        $Write\ whole\ page\ from\ buffer\ to\ file$
                        $bRecordUpdated \leftarrow true$
                        $return\ true$
                    end if
                end if
                $nRecordID \leftarrow nRecordID + 1$
            end while
        end if
        $nPageID \leftarrow nPageID + 1$
    end while
    if $bRecordUpdated\ is\ false$ then
        $Return\ false$
    end if
end procedure
```

Algorithm 7 Search For a Record

```
procedure SEARCHRECORD($TypeName$,$RecordKey$)
    $Open\ TypeName.dat\ file$
    $nPageID \leftarrow 0, bRecordFound \leftarrow false$
    while $((nPageID + 1) * 1024$ $is\ less\ than\ or\ equal\ to\ file\ size$ do
        $Read\ whole\ page\ whose\ ID\ is\ nPageID$
        if $(TotalSpace-FreeSpace)$ $of\ page\ is\ greater\ or\ equal\ to$ $(180+124)$ $bytes$ then
            $nRecordID \leftarrow 0$
            while $(nRecordID + 1) * 180$ $is\ less\ than\ or\ equal\ to$ $(1024 - 124)$ do
                if $isUsed\ flag\ of\ Record\ is\ 1$ then
                    if $Record\ primary\ key\ is\ same\ with\ input\ Record\ primary\ key$ then
                        $Print\ record\ information$
                        $bRecordFound \leftarrow true$
                        $return\ true$
                    end if
                end if
                $nRecordID \leftarrow nRecordID + 1$
            end while
        end if
        $nPageID \leftarrow nPageID + 1$
    end while
    if $bRecordUpdated\ is\ false$ then
        $Return\ false$
    end if
end procedure
```

Algorithm 8 List All Records of a Type

```
procedure ListRecords($TypeName$)
    $Open\ TypeName.dat\ file$
    $nPageID \leftarrow 0$
    while $((nPageID + 1) * 1024$ $is\ less\ than\ or\ equal\ to\ file\ size$ do
        $Read\ whole\ page\ whose\ ID\ is\ nPageID$
        if $(TotalSpace - FreeSpace)$ $of\ page\ is\ greater\ or\ equal\ to$ $(180 + 124)$ $bytes$ then
            $nRecordID \leftarrow 0$
            while $(nRecordID + 1) * 180$ $is\ less\ than\ or\ equal\ to$ $(1024 - 124)$ do
                if $isUsed\ flag\ of\ Record\ is\ 1$ then
                    $Print\ record\ information$
                end if
                $nRecordID \leftarrow nRecordID + 1$
            end while
        end if
        $nPageID \leftarrow nPageID + 1$
    end while
end procedure
```

5 Conclusion and Assessment

In this report, the design of Storage Manager was explained by expressing the design choices. As it was stated, the fix length record length was used. Moreover, the records and types were stored in the system catalog and data files as heap files. Sorted file or hash file didn't use.

The main advantage of using heap file is that it makes the programming and design easy.The other advantage is that it makes the delete operation much more faster. However, there are many disadvantage to use heap file. It increases the insert and update operations' complexity. In this design, the complexity of these operations directly proportional to the number of records or types in the storage. Nevertheless, it uses too much space even if it is not used because of the fact that after delete operation, we are not reorganizing the records or types after delete operation. Therefore, in this design, we can face with a scenario which page is allocated even if there is not any record inside it. Another disadvantage of this design is that it is not suitable the programs which use many threads which are trying to use our Storage Manager. Because we are not using any lock mechanism to handle this situation.

In the second phase of this project, this design will be implemented and some mistakes will be corrected such that although I assume that all fields are integer type, I allocate 16 bytes for every field. Instead of allocating 16 bytes, I will correct it as 4 byte.