BOGAZICI UNIVERSITY

CMPE321-SUMMER 2020

DESIGN OF A SIMPLE STORAGE MANAGEMENT SYSTEM FOR DMS

Halil BAYDAR
2017400297

Contents

1. Introduction

The aim of this document is giving the important information about my simple storage management system design which is communication tool between users and kept data in storage. This system is consisting of System Catalogue, File design, Page design, Record design ,which are shown by using tables/diagrams/figures, and this sytem does DML and DDL operations such as create/delete/list all Types and create/delete/search/etc Records ,which is explained in pseudo-code in following sections. And finally, I will evaluate my design and give feedback about it.

2. Assumptions & Constraints

2.1. Assumption

- All fields are integers.
- Type and field names are alphanumeric.
- User always enters valid input.
- If user creates a record but does not fill some of its fileds or all fields, these fields have default value such as “-1”.
- Primary key in records is unique and given by user.
- In the every file there is only one type.
- One char is 1 byte
- There is no limit of files’ size

2.2. Constraints

- Pages in which the data are organized only contains records.
- Min number of fields a type can have (≥ 3)
- Min length of a type name (≥ 8)
- Min length of a field name (≥ 8)
- Records are independent from index in which they are.
- The primary key of a record is assumed to be the value of the first field of that record.
- There must be different files.
- Files read pages one by one when it is needed, not whole at the same time even if all will be read.

3. Storage Structures

3.1. Record Format

The smallest unit of data is stored in record fields and all fields are in integer format. Records have two parts: Header and body. In the header part, there is primary key field that is unique for every record in the first position.During creation of a new type, the number of fields in body part are determined by user and the length of fields name must bigger than 8 lower than 15 characters. The fields in body part are bigger than 3 lower than 7.

- Primary key field = 15 byte
- Fields at most 14 byte
- Most total size of any record = 117 byte

Header Part	Filed 1	Field 2	……..	Field n

3.2. Page Format

There are two parts in pages: Page Header and Page Body. Page header has counter counting the number of record it has. Every page has a fixed size 2kB to decrease the necessity of accessing different pages. The default value of the # of records is zero. When the type is created, only one page is assigned and allocated as default to avoid wasting more space in disk and also to increase the productivity of the system. Every page can have at most 17 records.

- #of records = 2 byte

Number of records					
Record 1	Header Part	Filed 1	Field 2	……	Field n
Record 2	Header Part	Filed 1	Field 2	…..	Field n
⋮					
Record n	Header Part	Filed 1	Field 2	……..	Field n

3.3. File Format

Every file keeps information about pages, records and fields and all files keeps only one type.

Page 1			
Page 2			
⋮			
Page n			

3.4. System Catalogue

The goal of system catalogue is keeping all information about types. There is only one part which is body. The information about type name and field names are listed in the body of system catalog.

Type 1
Type 2
⋮
File n

4. Operations

4.1. DML Operations

4.1.1. Create a Record

```
Type ← User Input.
#of Fields ← User Input.
Record record = new Record(typename,key)
for field in record.fields:
    field.data ← User Input.
end
#this steps until this point is done by my storage manager system.
### In order to get first page, my storage manager send a request to file manager and file manager get the file address from storage manager. File manager give the disk manager the addres of file and page index. Disk manager takes given address to retrive the file and give a page of file to file manager. File manager looks for header part of it to read #of records and if this page has enough space, the file is given to storage manager to insert a new record, if not, page searching is going to until proper page is found, if not, new page is created.a###
//code

Myfile=file_manager(filename)
page=myfile.to_find_proper_page()#gives proper page to insert record
for i in range(len(page)):
     if page[i]==’\n’: #to find proper line in page
        page[i]=record
myfile.to_write_into_file_in_proper_page(page) #writting back this page
```

4.1.2. Delete a Record

Type ← User Input
RecordID ← User Input
In order to get first page, my storage manager send a request to file manager and file manager get the file address from the storage manager. File manager give address of file to disk manager. Disk manager takes given address retrives the file and gives pages of the file to file manager page by page. File manager returns pages to system catalog. And if wanted record is found, it is deleted and page is sent back to file and after that to disk manager.###
//code

```
myfile=file_manager(filename)
page=filem.to_delete_record() #gets pages one by one
for i in page:
  if i==wantedrecord:
     delete i  # delete it
```

4.1.3. Search For Record (by primary key)

Type ← User Input
RecordID ← User Input
In order to get first page, my storage manager send a request to file manager and file manager get the page address from storage manager. Disk manager takes given address and page number from file manager retrives pages from file and gives file manager it. At the end, storage manager takes pages one by one file manager and reads all details of pages of them. In first,this system looks for record key and if it is matched with user input record key, and it is printed,if not, searching is going on.###

```
for record in page: #iterates records in pages
    if recordID == givenID: #if primary key are correct
        for fields in record: #print records
            print(fields.name,fields.data)
        end
        break
    end
end
```

4.1.4. List all Records of a Type

Type ← User Input
In order to get first page, my storage manager send a request to file manager and file manager get the file pointer from storage manager. Disk manager takes given address retrives the pages one by one and gives them file manager. At the end, storage manager takes pages from file manager. And all page records of given type are printed.

```
for page in file: #iterate pages in file
    for record in page:  #iterates records in pages
            print(records)
    end
end
```

4.2. DDL Operations

4.2.1. Create a Type

```
TypeName ← User Input
Numberoffields  ←  User Input
#Create new field in the system catalog to push type into it
Type type= new Type(typename,numberoffields) #new type
for field in type.filelds: #fields name is filled by user input
      field.name ← User Input
end
system_catalog.push(type)
```

4.2.2. Delete a Type

```
TypeName ← User input
# My storage manager reads types in the system catalog
for type in system_catalog: #iterate files
          if typename == givenTypeName: #type names are mathced and then delete
            delete_type(type) #delete file related with given type
            delete(type) #delete type in systemcatalog
          end
      end
end
def delete_type(type):
     #in first iterates all files to delete related file with deleted type
     ### In order to get first file, my storage manager send a request to file manager and file manager get the file address from system catalog. Disk manager takes given address retrives file and the file is deleted by disk manager. ###
          for file in sytem_catalog:
              if filename == given_type_name: #if names are matched
                delete(file)
              end
          end
```

4.2.3. List all Types

```
for type in sytem_catalog: #iterate types in the system catalog
    print(file.page.typename)
```

5. Conclusions & Assessment

In this documantation, the details of my storage management system was explined. To minimize used storage space, I did not keep records size fixed and their size is determined by given fields number by user but the time comlexity in the process of creating new record is a bit high. This system keeps and insert pages and records into disk in unordered list. In the creation of new record, this system looks for emty space in page, if there is no space,new pages are created. The system reads pages one by one not at the same time. In th system, files are created with given type name so all

names of them is same type names. These names are used for pointer pointing files in implementation.I think that my system works efficiently even in big data, there is no any issue in processing all of data.

The differences my storage management system implementation from the first homework are that there is no system catalog header and there is only #of records field in the header part of pages because I think all other fields are reduntant. And also one page can has at most 17 records. If one record is deleted, the line in which record was is kept as empty line instead of deleting it.