CMPE321, 2018-2019-2, Database Storage Manager System, Ömer Faruk Özdemir 2016400048

Ömer Faruk Özdemir

March 10, 2019

Contents

1 Section One

1.1 Introduction

In this project I need to design a simple database storage manager system. I need to decide how to hold the records, design system catalogue, pages and records. I also need to create algorithms for DDL and DML operations.

Since this is a simple storage manager, I will not be doing error checking, like wrong input or duplicate records. However, if it is wanted they are easy to design or implement, since the main idea is done.

1.2 Assumptions & Constraints

- There is no null value for any record.
- Every record is integer.
- There is no bad entry, so I will not be doing replicate or wrong input checks.
- Since this is a simple storage manager I will pick max number of field in a type as 10.
- Maximum name length of a type or field name is 10 as well.
- Maximum number of types is 10 as well. There is no need to hold system catalogue in pages.
- I will hold a specific type's records in same pages and same files.
- Files will be txts and pages will be maximum 128 lines in a txt.
- Records will be single lines in the txt, so a page will hold 128 records at maximum.
- If we take every entry as 1 byte, a page will be 1280 bytes at most, meaning 1KB.
- A page will be 8KB at most.
- When a page needs to be loaded to ram, only appropriate lines will be loaded, not whole txt.
- Type name, number of fields, field names, number of files, file names, number of lines in the file will be held in system catalogue.
- Every file will hold maximum of 8 pages. After that, a new file will be created. And file names will be held in the system catalogue.
- When a file becomes empty it will be deleted from disk, and its name will be deleted from system catalogue.

- File names will be like this for a type named Dog: Dog1, Dog2, Dog3, Dog4, Dog5...

1.3 Storage Structures

In theory, database takes necessary information from the system catalogue, for example a page's, file manager will request that page from the disk manager, and disk manager will retrieve the page from the disk and give it to file manager.

- System Catalogue:
  - System catalogue will hold types, that type's name, number of fields, fields' names, number of files, names of files, number of lines in those files. That line number can give record number or page number in the file.
  - System Catalogue will hold types line by line.
  - Maximum page number in a file is 8, and at maximum every page has 128 records. Since every page is 128 lines at max.

  - Photo of a system catalogue –

  Couldn't add to latex please look to ReadMe

- Files
  - Files are in txt format, records are lines in that txt, and pages are blocks of 128 lines.
  - Files are made of pages. My max page number is 8 in a file. I will hold all records of a specific type in a file named of that type.
  - File names will be "typename"+"1", "typename"+"2", "typename"+"3"...
  - When every file is full, a new file will be created with increasing number of the last file's name.
  - When a file gets empty, meaning there is 0 line in that file, this can be accessed from the system catalogue, delete the file from both disk and its name from system catalogue.
- Pages
  - Pages are made of records.
  - My page limit is 128 lines in a txt file, so every page can contain 128 records at maximum.

  - There is no page header.

- Records
  - Records are 1 line. It is integers with spaces, with the same order as fields in the system catalogue.
  - There is no record header.

- Photo of a file –
Couldn't add to latex please look to ReadMe

1.4 Operations

- DDL Operations
  - Create a type: Append type name and its field names to system catalogue.
  - Delete a type: Reach to that type from system catalogue, get all file names about that type, delete all files about that type, then delete information related to that type from system catalogue. Note that there is no need to load pages or files to ram, just will delete files from the disk.
  - List all types: Read type names from system catalogue, sort them alphabetically, then print.

- DML Operations
  - Create a record: Find that types file names from the system catalogue, if there is no file name create a file with name "typename"+1 , else start from first file, if file is not full append the record to the last page, else continue with the next file, if all files are full create a new file, write that file name to system catalogue, increase file number of that type by 1 in system catalogue, append the record to new file's first page.
  - Delete a record: Find that type's file names from the system catalogue, start from the first file, going page by page, scan primary values, find the correct record, delete the line of that record, decrease line number of that file from system catalogue, if file is empty after deletion, delete the file from disk and delete its name from system catalogue. Decrease file number of that type by one from system catalogue.
  - Search for a record (by primary key): Find that type's file names from the system catalogue, start from the first file, going page by page, scan primary values, find the correct record, and get that line. Print it with field names from system catalogue.

– List all records of a type: Reach to the system catalogue; find that type's file names, going page by page print all records.

2 Algorithms

Note: Accessing files page by page means just loading corresponding lines in that txt.

Algorithm 1 Create a type

1: Get user input, which contains "typename" "field1name" "field2name"...
2: Count number of words.
3: numberOfFields = counted number of words -1.
4: Append it to system catalogue's txt file like this, "typename" numberOfFields "field1name" "field2name" ... 0

Algorithm 2 Delete a type

1: Search system catalogue line by line
2: Reach to that type's line with the typeName(note that first word in every line is typeName).
3: Get fileNames from there.
4: Delete every file.
5: Delete the line of that type from system catalogue.

Algorithm 3 List all types

1: Search system catalogue line by line.
2: Print typenames (note that first word in every line is typeName).

Algorithm 4 Create a record

1: Take input from the user. As “typename” “field1” “field2” ...
2: Find that typename’s line from the system catalogue (note that first word in every line is typeName).
3: Get fileNumber.
4: **if** fileNumber==0 **then**
5: Create a new txt file with name “typename”+1
6: Increase fileNumber by 1 in the system catalogue.
7: **end if**
8: Scan the fileNames and lineNumbers one by one
9: **if** every file has 128*8 line **then**
10: Create a new file with increasing number from lastfile’s number.
11: Increase fileNumber by 1, write that file’s name to the system catalogue
12: **end if**
13: Append the record to first file with line less than 128*8’s last page as “field1” “field2” ..
14: Increase that file’s lineNumber by 1 in the system catalogue.

Algorithm 5 Delete a record

1: Scan system catalogue
2: Reach to the line of that type
3: Get the file names one by one(since there is no error input, there should be at least 1 files of that type)
4: Scan file page by page, scan pages line by line
5: **if** first word in the line == givenPrimaryValue **then**
6: Delete that line
7: Decrease number of lines of that file from system catalogue by 1.
8: **end if**
9: **if** number of lines==0 **then**
10: Delete the file from disk, delete file name and 0 from the system catalogue, decrease file number by 1 from system catalogue
11: **end if**

Algorithm 6 Search for a record (by primary key)

1: Scan system catalogue
2: Reach to that type’s line
3: Get that type’s file names.
4: Access to files page by page, scan pages line by line.
5: **if** first word in the line == givenPrimaryValue **then**
6: Copy that line, print it with field names from that type’s line in system catalogue.
7: **end if**

Algorithm 7 List all records of a type

1: Scan system catalogue.
2: Get that type's file names.
3: Access to files page by page, copy each line.
4: Print one by one with field names from that type's line in system catalogue.

3 Conclusion & Assessment

- This was a simple storage manager. For a more advanced one, number limitations can be increased. Maximum field number, name lengths, type number can be increased.
- Fields just support integer. For a more advanced design other types can be supported as well.
- Since this is a simple design compression and efficiency are not real concerns. All records of a type are held in same files and pages. There is no dynamic relocation of pages or files. Records stay in the same place. For efficiency and compression some other configurations can be done.
- There is no security concern for this design. Nothing is encrypted, everything is public in txt files.
- This not an elegant design, if it was created for usage in professional life it would need to be more professional. It is a simple design, and easy to understand.