CMPE 321 SPRING PROJECT 1

Design a Storage Manager System

Şadi Uysal 2015400162

Bogazici University

1 Introduction

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.

In this project, I am going to implement a storage manager system which I designed in the first project.I did some cool changes from my design and i explained more detail in the conclusion section. In the storage manager, I am going to implement System Catalog,records,pages and after that, implement some algorithms to create and manipulate data via DML,DDL operations.

2 Assumptions & Constraints

- User always enters valid input and fields are always integers, though field and type names can be alphanumeric.
- A disk manager already exists that is able to fetch the necessary pages when addressed.
- A char equal 1 byte.
- Data must be organized in pages and pages must contain records.
- Maximum file size is 4096 KByte.

- System must not load the whole file to RAM when it is needed. Instead, it must read a file page by page.

2.1 Record

- Record header has a state byte which is first byte that contains information about the state of the record such as full or empty.
- Every record has fixed number of fields that could be change from type to type.
- Maximum number of fields is 6 for this project.
- Every non-empty record has a primary key which is first field of the record.
- Every record could be searched by primary key which is unique for each.
- Each record field size 8 bytes.

2.2 Page

- Maximum page size is fixed and 1024 bytes.
- Every page header size is 9 bytes.
- Page header consists of page ID, number of records count,first available slot for a record.
- Page ID is first 3 byte for each page.
- Number of records 3 byte for each page.

- First available slot for a record is 3 byte for each page.
- Each page consists of just one type's data.

2.3 System Catalog

- System catalog file is named as catalog.dat.
- Every type must have a name that can be between 2 and 10 bytes.
- Max number of fields a type can has is 6.
- Field names can be between 1-8 bytes.
- In catalog.dat, a type declaration has 1 byte for the status of the type such as empty or full which is first byte.
- After that it contains type name that could be between 2-10 bytes.
- It also contains number of fields which is 1 byte.
- And the last part names as many as number of fields.

2.4 Index File

- Index file consists of primary key of the record and the pageID.
- First 8 byte is primary key,9th byte is $ sign and the other 3 byte pageID.
- Every index entry is 12 byte as mentioned above.

3 Storage Structures

3.1 Record

Header	At most 6 field
State Byte	Field Value
1 Byte	8 Bytes

3.2 System Catalog

State	Type Name	#of Fields	Names
1Byte	10 Bytes	1 Byte	8Bytes
		max 6	

3.3 Index File

Primary Key	$	pageID
8 Bytes	1 Byte	3 Bytes

3.4 Page

Header			at most 20 record
PageID	#of Records	1th empty slot	Records
3 Bytes	3 Bytes	3 Bytes	(49Bytes)*(nofR)

4 Operations

4.1 DDL Operations

4.1.1 Create a Type

Algorithm 1 Create a Type

Require: $typeName$,$nofFields$,$fieldSizes$,$fieldNames$
  if $typeName$ does not exist in the catalog.dat **then**
    $i \leftarrow 0$
    while State byte of the type entry i =Full **do**
      increment i++
    end while
    //New type entry location = i
    State byte of the entry[i]=Full
    Name of the type = $typeName$
    Number of fields's byte=$nofFields$
    for j=0 to $nofFields$ **do**
      Field size byte j=$fieldSizes$[j]
      Field Name bytes j=$fieldNames$[j]
    end for
  end if

4.1.2 Delete a Type

Algorithm 2 Delete a Type

Require: *typeName*
  $i \leftarrow 0$
  while State byte of the type entry i =Full **do**
    if Type entry[i]'s TypeName Bytes=*typeName* **then**
      Entry[i]'s state byte=Empty
      BREAK
    end if
    increment i++
  end while

4.1.3 List all Types

Algorithm 3 List all Types

  $i \leftarrow 0$
  while Catalog.dat does not finished **do**
    if State byte of the type entry i =Full **then**
      Print Type entry[i]'s *typeName*
    end if
    increment i++
  end while

4.2 DML Operations

4.2.1 Create a record

Algorithm 4 Create a record

Require: $fieldsOfRecord$
  if Type's first page does not exist **then**
    Create new page with $pageID$,nofRecords=0,firstemptyslot=0
  end if
  Go to type's first page assign its pageID to $pageID$
  while $pageID$ exists **do**
    if PageID's 5th byte(first empty slot) is not equal to *null* **then**
      Go to first empty record slot
      Set State byte of the record=Full
      for i=0 to Type's nofFields **do**
        Record's field i= $fieldsofRecord$[i]
      end for
      increment page's 4th byte(number of records) by 1
      find new first empty record slot and update 5th byte(first empty slot)
      BREAK
    else
      if next page does not exist **then**
        Create new page with $newpageID$,nofRecords=0,firstemptyslot=0
        Set $pageID$'s next=$newpageID$
      end if
      $pageID$ =next $pageID$
    end if
  end while

4.2.2 Delete a Record

Algorithm 5 Delete a Record

Require: *primarykey*
call function Search for a record (*primarykey*)
get the Record
set record's state byte to empty
update corresponding pageID's nofRecords byte and first empty slot byte

4.2.3 Search for a record(by primary key)

Algorithm 6 Search for a record(by primary key)

Require: *primaryKey*
if Type's first page does not exist **then**
print "No such record exists.
end if
Go to type's first page assign its pageID to *pageID*
while *pageID* exists **do**
for *i*=0 to numberOfRecords **do**
Go to record i
if record[i]'s state byte=full **then**
if record[i]'s first field=*primarykey* **then**
Return record[i]
BREAK
end if
end if
increment *i*++
end for
if next page does not exist **then**
print "No such record exists.
return *null*
end if
pageID =next *pageID*
end while

4.2.4 List all records of a type

Algorithm 7 List all records of a type

Require: *typeName*
  if Type's first page does not exist **then**
    print "No such record exists.
  end if
  Go to type's first page assign its pageID to *pageID*
  while *pageID* exists **do**
    for *i*=0 to numberOfRecords **do**
      Go to record i
      if record[i]'s state byte=full **then**
        Print record[i]
      end if
      increment *i*++
    end for
    if next page does not exist **then**
      BREAK
    end if
    pageID =next *pageID*
  end while

5 Conclusion

It was quite challenging project but i learned a lot. Especially i learned a lot about writing,reading into files, I/O Operations.It takes time and effort but it wasn't boring.

I realize it could be better to change some design choices so i changed some of it.My page header was just 5 bytes and i changed it to 9 bytes so that i can easily use empty slot and number of records in operations.Moreover, Field sizes, field name sizes, maximum number of fields, type name size were the other changes. I also added index file and it ease my search,update algorithms much. I removed field size fields from System catalog since it was fixed for this project.

Designing and implementing such a system was not easy but as i said it wasn't that much boring and i sit on the computer like 8 hours. But it would be better if we had more time to do it.

All in all, it is funny to study on database management and i am looking forward to learn more and practicing more with these projects even if i tried hard and got problems while doing it. I think it will get better day by day.