Cmpe321 Project Halo Report

Erencan Uysal - 2017400069
Muhammed Göktepe - 2017400162
Emir Hıfsı Öztoprak - 2017400060

July 3, 2021

Contents

1 Introduction **3**

2 Assumptions & Constraints **3**
2.1 Assumptions 3
2.2 Constraints 3

3 Storage Structures **4**
3.1 System Catalogue 4
3.2 Record 4
3.3 Page 5
3.4 File 5

4 Operations **6**
4.1 HALO Definition Language Operations 6
4.2 HALO Management Language Operations 7

5 Conclusion & Assessment **9**

1 Introduction

In this project we designed a database storage management system. This system has records, pages, files and headers for these structures. These structures help us to efficiently handle storage and search operations. This system supports create, delete, inherit and list operations for type; create, delete, search, update, list and filter operations for records.

2 Assumptions & Constraints

2.1 Assumptions

- Each field has 20 byte space.
- Each page has 2 kb (2048 byte) space.
- A file has 8 pages in it.
- All fields are alphanumeric.
- User always enters valid input.
- Each field must have a value. So there is no null value for any field.
- We assumed that the inputs are given in correct form.
- We assumed the input lines are given line by line.
- We assumed the condition for the filter operations is given in "age<20" form with no spaces in between.

2.2 Constraints

1. The data must be organized in pages and pages must contain records. (We can not implement this in this project due to limited time.)
2. You are not allowed to store all pages in the same file and a file must contain multiple pages. This means that your system must be able to create new files as HALO grows. Moreover, when a file becomes free due to deletions, that file must be deleted.(We can not implement this in this project due to limited time.)
3. Although a file contains multiple pages, it must read page by page when it is needed. Loading the whole file to RAM is not allowed.(We can not implement this in this project due to limited time.)
4. The first attribute of all types in HALO software must be a string type, named as "planet" and its value for all records must be "E226 - S187".

5. The primary key of a record should be assumed to be the value of the second field of that record.

6. Records in the files should be stored in descending order according to their primary keys.

3 Storage Structures

3.1 System Catalogue

System catalogue gives us necessary information about main situation of our database storage system. We used a .txt file as system catalogue. We used this file frequently in control mechanism of our system. For example we controlled type names whether they already exist before creating a new type. We add information related to created types to our system catalogue. We did not add planet and record id fields to our system catalogue because they are common in all types. System catalogue holds:

- Types
- Number of fields for each type
- Names of fields for each type
- Number of created files for each type

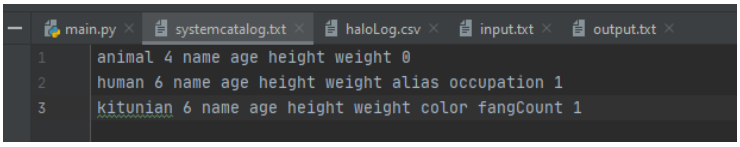

Figure 1: System catalogue

System Catalogue	Field 1	Field...	File Count
Type 1	*name*	*age*	0
Type 2	*name*	*weight*	5
Type 3	*password*	*age*	1

Table 1: System Catalogue

3.2 Record

For each type, we should make calculations. We are assuming each field is 20 byte. Also we have a record header for each record. We stored record header in 20 bytes. So, each record occupies (number of field + 1) times 20 bytes.If a

type has 4 fields, then we should have 20*5=100 bytes for each record. Record header holds record id. We can easily sort records in this way.

Record Header	**Field 1**	**Field 2**	**Field ...**
88	$E226 - S187$	88	$John$
65	$E226 - S187$	65	$Jacob$
55	$E226 - S187$	55	$Mark$

Table 2: Record

3.3 Page

Page is the fundamental structure of storage system. Pages holds records. Each page has 2 kb (2048 byte) storage space for records. We have one type of records in each page, so we can easily calculate how much record can fit in one page. Record per page differs based on number of fields for each type. Page header is 20 byte, so we have 2028 bytes for records. Firstly we should calculate space for one record than divide 2028 by it. Page header holds lowest record id and number of empty spaces in this page. We can easily traverse pages in this way.

Page Header:	**Lowest : 55**	**Empty spaces : 5**	
Record header : 88	$E226 - S187$	88	$John$
Record header : 65	$E226 - S187$	65	$Jacob$
Record header : 55	$E226 - S187$	55	$Mark$

Table 3: Page

3.4 File

Files have 8 pages. File header holds lowest record id and number of pages in this file. We can search pages in a for loop with the help of this number. We can also easily see whether a record lies in boundaries of this file with the help of lowest record id.

File Header:	**Lowest : 55**	**Page Count : 2**	
Page header :	$Lowest : 55$	$Emptyspaces : 5$	
Record header : 65	$E226 - S187$	65	$Jacob$
Record header : 55	$E226 - S187$	55	$Frodo$
Page header :	$Lowest : 24$	$Emptyspaces : 2$	
Record header : 44	$E226 - S187$	445	$Harry$
Record header : 38	$E226 - S187$	38	$Mark$
Record header : 35	$E226 - S187$	35	Ron
Record header : 29	$E226 - S187$	29	$Bilbo$
Record header : 24	$E226 - S187$	24	Joe

Table 4: File

4 Operations

DDL operations:

- Create Type
- Delete Type
- Inherit Type
- List Type

DML operations:

- Create Record
- Delete Record
- Update Record
- List Record
- Search Record
- Filter Record

4.1 HALO Definition Language Operations

Algorithm 1 Create Type

1. Get input from user, which contains "create type" operation and field names
2. Count field names from input
3. Control if given number is equal to number of fields. Otherwise print failure to log file.
4. Check systemcatalog.txt file if given type name exist.
5. If it exist print failure to log file.
6. If it does not exist append this type and its fields and 0 as number of files this type has to systemcatalog.txt file and print success to log file.

Algorithm 2 Delete Type

1. Get input from user, which contains "delete type" operation.
2. Check systemcatalog.txt file if given type name exist.
3. If it exist delete that type from systemcatalog.txt file and delete all files of that type and print success to log file.
4. If it does not exist print failure to log file.

Algorithm 3 Inherit Type

1. Get input from user, which contains "inherit type" operation and new type name, old type name and new types' field names.

2. Check systemcatalog.txt file if given old type name exist.

3. If it exist move to step 5.

4. If it does not exist print failure to log file.

5. Check systemcatalog.txt file if given new type name exist.

6. If it exist print failure to log file.

7. If it does not exist append this type and its fields and 0 as number of files this type has to systemcatalog.txt file and print success to log file.

Algorithm 4 List Type

1. Get input from user, which contains "list type" operation.

2. Read systemcatalog.txt file and print typename if there exist any type.

4.2 HALO Management Language Operations

Algorithm 5 Create Record

1. Get input from user, which contains "create record" operation and typename and field values

2. Count field values from input

3. Control if given number is equal to number of fields. Otherwise print failure to log file.

4. Read file count from systemcatalog.txt file for given type name.

5. Starting from the last created file check file header for lowest record id in this file. If new record id is smaller, check previous file. Stop if you are in first file.

6. Starting from first page in this file, check page header for lowest record id in this file. If new record id is smaller, check next page. Stop if you are in last page in this file.

7. Check if given record id exist in this page.

8. If it does not exist append new record to this page and print success to log file.

9. If this page is full, append record and sort again. Then put lowest record to new page.

10. If file is full, put lowest record in file to another file recursively.

11. If it already exist print failure to log file.

Algorithm 6 Delete Record

1. Get input from user, which contains "delete record" operation and typename and record id

2. Read file count from systemcatalog.txt file for given type name.

3.Starting from the last created file check file header for lowest record id in this file. If new record id is smaller, check previous file. Stop if you are in first file.

4. Starting from first page in this file, check page header for lowest record id in this file. If new record id is smaller, check next page. Stop if you are in last page in this file.

5. Check if given record id exist in this page.

6. If it does not exist print failure to log file.

7. If it exist delete that record from this file and move all records below the deleted record up one line. Print success to log file.

Algorithm 7 List Record

1. Get input from user, which contains "list record" operation and typename.

2. Read file count from systemcatalog.txt file for given type name.

3. If given type does not exist in systemcatalog.txt print failure to log file.

4. If it exist read all files of given type and print all records to output file. Print success to log file.

Algorithm 8 Search Record

1. Get input from user, which contains "search record" operation and typename and record id

2. Read file count from systemcatalog.txt file for given type name.

3.Starting from the last created file check file header for lowest record id in this file. If new record id is smaller, check previous file. Stop if you are in first file.

5. Starting from first page in this file, check page header for lowest record id in this file. If new record id is smaller, check next page. Stop if you are in last page in this file.

6. Check if given record id exist in this page.

4. Check if given record id exist in this file.

5. If it does not exist print failure to log file.

6. If it exist print that record to output file. Print success to log file.

Algorithm 9 Update Record

1. Get input from user, which contains "update record" operation and typename and record id

2. Read file count from systemcatalog.txt file for given type name.

3.Starting from the last created file check file header for lowest record id in this file. If new record id is smaller, check previous file. Stop if you are in first file.

4. Starting from first page in this file, check page header for lowest record id in this file. If new record id is smaller, check next page. Stop if you are in last page in this file.

4. Check if given record id exist in this page.

5. If it does not exist print failure to log file.

6. If it exist update that record. Print success to log file.

Algorithm 10 Filter Record

1. Get input from user, which contains "filter record" operation and typename and condition.

2. Read file count from systemcatalog.txt file for given type name.

3.Starting from the last created file check all records and if given condition holds for that record print it to the output file.

4. Print success to log file.

5 Conclusion & Assessment

We tried to implement the design we explained above. We used bytearrays firstly. Unfortunately we could not implement all system like this. We had limited time, so we implemented a working and simpler code. We can not implement page design.

We implemented system catalog and type systems correctly. But we had to implement record functions in a more basic way due to time constraint.

Our system is working correctly but it is not the way we described above.