Project 4: Project HALO

CMPE 321, Introduction to Database Systems, Spring 2021

Due: **July 4, Sunday, 11:55 AM**

Please ask the project related questions in the Moodle forum!

1 Introduction

In the era of information and connection, collecting and storing data have become two of the most critical operations that must be handled delicately. Accumulation and manipulation of all those data are achieved through digital storage units. In 321Corp, our true aim is to define the very foundation of digital storage. There's nothing that can't become digital, and we challenge ourselves each day with the unpredictability of life. Therefore, we hereby announce the cutting edge digital storage management system, HALO, Highly Attributed Logical Object. It is quite a breakthrough and it will represent its worth day by day, year by year.
Found among the salvaged blueprints of *HALO* software

BRIEFING: Project *HALO*

After the solar storms, there was not a single machine intact and functioning in our LifeDome (planet), thus all digital knowledge curated and stored starting from the 20th century has been lost. Salvaging the uncompromising parts was not easy, yet we managed to collect enough material to sustain and rebuild our primary systems. Now, it is time to reestablish data management. We, engineers of 321Corp, are the only hope of all people residing in our LifeDome that can revive the digital era and reconnect with the Earth once more. To do that, we need to establish a *HALO* center to store all data transferred from the primary systems. Surviving engineers of the Universal Space Station are taking care of the physical infrastructure of the *HALO* instances, hence we need to implement and launch *HALO* software. Luckily, several parts of the *HALO* software blueprints that survived the solar storms have been found. We can use the instructions and constraints in those blueprints to have a starting point. Improvisation is allowed if necessary situations occur. Even though we don't have much time, we have challenging tasks to accomplish. **Let the *Project HALO* commence!**

...

You are one of the residents of planet $E226 - S187$ in the Tykhan System in Zenith, which is a galaxy very distant from our solar system. Because you participated in one of the trial expeditions performed by the United Earth Confederation, you are currently living in $E226 - S187$, which is called as LifeDome among the residents. Your skills of engineering are second to none, thus you are selected to create the *HALO* software to establish a *HALO* center for the society. The previous establishment was destroyed, thus your society needs your experience and skills once more to establish an integrated system of information, which consists of retrieval, process, and storage. You are in charge of a storage division, named *Project HALO* and **you will design and implement the *HALO* software** by defining its properties **according to the constraints and instructions**. The briefing above has been given by the chief engineer of the 321Corp and you have **21 Earth Days** to finish the project. **Since this is a mission-critical project, there's no chance of postponing and the deadline is STRICT. You may select one 321Corp engineer to cooperate with you during the project.**

2 Phase Alpha: Design

In this phase, you are expected to design the *HALO*. The design shall have a system catalog for storing the metadata and data storage units (files, pages, and records) for storing the actual data. The system design must support the following operations:

***HALO* Definition Language Operations**

- Create a type
- Delete a type
- Inherit a type
- List all types

***HALO* Authentication Language Operations**

- Login to *HALO*
- Logout from *HALO*
- Register in *HALO*

***HALO* Management Language Operations**

- Create a record
- Delete a record
- Search for a record (by primary key)
- Update a record (by primary key)
- List all records of a type
- Filter records by one of the attributes of a type

As the chief engineer in 321Corp explained in the briefing, you need to **obey the instructions** while designing the *HALO* system. Thus, the instructions shall ***include***, but are not limited to:

- Define page size (Something between 2KB and 3KBs is optimal)
- Define file size (Should contain reasonable amount of pages)
- Define what information to store in your page headers and record headers
- Define number of fields a type can have (≥ 6)
- Define length of a type name (≥ 12)
- Define length of a field name (≥ 12)

Assumptions:

- All fields shall be *alphanumeric*. Also, type and field names shall be alphanumeric.
- User always enters valid input. There will be no non-alphanumeric character inside the test cases.
- The hardware of *HALO* center and *HALO* instances will be built according to the blueprints, thus you do not need to consider the *HALO* physical storage controller to interact with the storage units.

You can improvise by making further assumptions, as long as they do **not conflict with the base assumptions** and they are **explained in the report CLEARLY**.

Constraints:

- The data *must* be organized in pages and pages *must* contain records. So, you *must* clearly explain your page and record structure in your report.
- 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.
- 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.
- 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 "$E226 - S187$".
- The primary key of a record should be assumed to be the value of the **second** field of that record.
- Records in the files should be stored in descending order according to their primary keys.

3 Phase Beta: Implementation

In this phase, you are expected to implement the *HALO* software that you have designed in the **Phase Alpha**. The implementation must support the ***HALO* Authentication Language Operations**, ***HALO* Definition Language Operations** and the ***HALO* Management Language Operations**.

4 Input/Output

Submissions will be graded **automatically** for various test cases. The allowed programming languages are **Java** and **Python3**. The input and output file names will be given as arguments to your executable program. For each language, the commands that will be executed are as follows:

Java

```
javac 2017400XXX/src/*.java
java -classpath 2017400XXX/src/haloSoftare inputFile outputFile
```

Python3

```
python3 2015400XXX/src/haloSoftware.py inputFile outputFile
```

Your directory must have the structure as shown below and the name of your main file must be **haloSoftware**.

```
2017400XXX.pdf
README.md
src
    haloSoftware.java
    temp1.java
    temp2.java
    ..
    ..
    ..
```

```
2017400XXX.pdf
README.md
src
    haloSoftware.py
    temp1.py
    temp2.py
    ..
    ..
    ..
```

Input file contains a list of operations. Each line corresponds to an operation. The format for each operation is as follows:

HALO Authentication Language Operations

Operation	Input Format	Output Format
Register	register user <user-name><password><password-repeat>	None
Login	login <user-name> <password>	None
Logout	logout	None

HALO Definition Language Operations

Operation	Input Format	Output Format
Create	create type <type-name><number-of-fields><field1-name><field2-name>...	None
Delete	delete type <type-name>	None
Inherit	inherit type <target-type-name> <source-type-name> <additional-field1><additional-field2>...	None
List	list type	<type1-name> <type2-name> ...

HALO Management Language Operations

Operation	Input Format	Output Format
Create	create record <type-name><field1-value><field2-value>...	None
Delete	delete record <type-name><primary-key>	None
Update	update record <type-name><primary-key><field2-value><field3-value>...	None
Search	search record <type-name><primary-key>	<field1-value><field2-value>...
List	list record <type-name>	<record1-field1-value><record1-field2-value>... <record2-field1-value><record2-field2-value>... ...
Filter	filter record <type-name><condition>	<record1-field1-value><record1-field2-value>... <record2-field1-value><record2-field2-value>... ...

Sample Inputs & Outputs

Bonus test case is provided below. Table 1 shows the content of the sample input file and the output file. Table 2 shows the logs of corresponding operations. Please check Drive link for normal and bonus example test cases.

Sample Input File	Sample Output File
login useralpha e226s187Xeno	E226-S187 3 MarkWebbler 27 178 81 Shadow engineer
register user useralpha X3n0m3R X3n0m3R	
login useralpha X3n0m3R	
create type animal 4 name age height weight	
inherit type human animal alias occupation	
create record human 3 MarkWebbler 27 178 81 Shadow engineer	
list record human	
list record animal	
logout	
list record human	
list type	

Table 1: Bonus Test Case - Input & Output

Log File
useralpha,1623001281,login,failure
null,1623001291,register user useralpha,success
useralpha,1623001301,login,success
useralpha,1623001321,create type animal 4 name age weight height,success
useralpha,1623001361,inherit type human animal alias occupation,success
useralpha,1623001381,create record human 3 MarkWebbler 27 178 81 Shadow engineer,success
useralpha,1623001391,list record human,success
useralpha,1623001401,list record animal,failure
useralpha,1623001421,logout,successful
null,1623001441,list record human,failure
null,1623001451,list type,failure

Table 2: Bonus Test Case - Log File

5 Implementation Instructions

- With the bonus case: There can only be one user logged in to *HALO* software at any time. Also, any definition and management operation can **only be performed while being logged in.** Therefore, any operation **without a login to *HALO*** should be logged with a username **null**.
- Without the bonus case: Consider there is only one user, whose name is *Admin*, currently logged in to the system. Therefore, any performed operation should be logged with a username **admin**.
- Filter operation has a condition part, which can only filter the integer attributes and only less than $<$, greater than $>$, and equal $=$ operators can be used to create a condition.
- Search, update and deletion of records shall always be done by **primary key**.
- When a type is deleted, all records of that type must be deleted.
- Types must be listed by ascending type names. Note that the built-in sort functions do sorting case-sensitively by default. This means that the elements starting with uppercase letters come first in the order.
- Test cases are not necessarily independent of each other. So, if the type Human is created in test case 1, it should be accessible in test case N.
- You can assume that type names, field names, and values cannot be longer than 20 characters.
- You can assume that the maximum number of fields cannot be greater than 12.
- Consider possible leading and trailing spaces in input lines. Write robust codes that handle such situations as well.

- Make sure `delete type` command works as expected, it will be used between independent test cases.
- *HALO* software must log all authentication, definition, and management operations into a CSV file (namely haloLog.csv) which consists of 4 columns, namely username, occurrence, operation, and status. Username is the name of the current user, the occurrence is the string form of UNIX time, the operation is the string form of the whole operation line prompted in the console, status is the result of the operation. Log file must be persistent and never deleted when the *HALO* software is either stopped or restarted.
- Status of an operation is decided whether or not the result of the operation is empty. For the listing, searching, and filtering operations, if the output of the operation is empty, it must be considered as **failure** and logged accordingly.

6 Report & Grading

You are expected to submit a report *written in LaTeX* that contains the sections below. Also, you have to provide a README file including your instructions to run ***HALO*** *written in Markdown*

1. **Title Page:** Write course name, semester, assignment title, your name, and student number, and the name and student number of your colleague if exists.
2. **Introduction:** Briefly describe the project in your own words.
3. **Assumptions & Constraints:** Clearly specify your assumptions and constraints of the system in an itemized or tabular format.
4. **Storage Structures:** Explain your system catalog, page design, page header, record header, etc. with tables/diagrams/figures.
5. **Operations:** Explain your ***HALO* Authentication, Definition, and Management Language Operations**. Refer to the **related files and functions** to improve the explanation.
6. **R & D Discussions:** Discuss possible improvements and explain how the *HALO* Software should be updated to establish a *HALO* cluster around the LifeDome, composed of multiple *HALO* centers.
   - What would be your solution to deal with mission-critical distributed operations executed on any *HALO* center?
   - What would be your primary concern regarding the distributed operations?

   Explain clearly and concisely.
7. **Conclusions & Assessment:** Evaluate your design, considering its ups and downs.

(Tentative) Grading weights are as follows:

- Test cases: 50%
- Auto-Runnability: 10%
- README instructions to run ***HALO***: 5%
- Report: 35%

7 Submission

This project can be implemented either individually or as a team of two people. The submission must include your report, source code, .tex file, and a README file that describes how to run your code. If your file size is over the Moodle upload limit, submit a link (Drive, Dropbox, etc.) for downloading your project. Place all the required files into a folder named with the student IDs of the team members separated by an underscore (e.g. 2017400200_2018700120). Zip the folder for submission and name the `.zip` file with the same name. Submit the `.zip` file through Moodle until the deadline. **Any other submission method and late submissions are not allowed**. Each group should submit **one** .zip file. Note that your submissions will be inspected for plagiarism with previous years' submissions as well as this year's. **Any sign of plagiarism will be penalized.**