Project 4: Project Horadrim

CMPE 321, Introduction to Database Systems, Spring 2022

Due: **May 26, Thursday, 23:59**

Please ask the project related questions in the Moodle forum!

1 Introduction

Nearly three hundred years ago, it came to be known that the three Prime Evils of the Burning Hells had mysteriously come to our world. The Three Brothers ravaged the lands of the East for decades, while humanity was left trembling in their wake. Our order - the *Horadrim* - was founded by a group of secretive Magi to hunt down and capture the three Evils once and for all. The original *Horadrim* captured two of the Three within powerful artifacts known as Soulstones and buried them deep beneath the desolate Eastern sands. The third Evil escaped capture and fled to the West with many of the *Horadrim* in pursuit. The third Evil - known as Diablo, the Lord of Terror - was eventually captured, his essence set in a Soulstone and buried within Tristram Cathedral. With the threat seemingly gone, and no great quest to drive them, the *Horadrim*'s numbers dwindled. The order eventually faded into history. The last known descendant of the *Horadrim* was Deckard Cain, a mere scholar with no experience in magic or battle. He would have passed his teachings on to anyone but there was nobody that would *sit a while and listen*. Thus he wrote down the battles he witnessed, the monsters he feared and the pain he suffered and everything he inherited from the past *Horadrim* members.[1]

. . .

Today, as the last living descendants of the *Horadrim*, it is up to you and your partner to put the pieces together that are left by Deckard Cain to uncover the stories of forgotten heroes, fights between demons and angels, menacing monsters, lost weapons and armors. *Horadrim* needs your experience and skills to establish an integrated system of information, which consists of retrieval, processing, and storage. You will design and implement the *Horadrim* software by defining its properties according to the constraints and instructions set by Deckard Cain. Let's cleanse this wilderness.

2 Chapter One: Design

In this phase, you are expected to design the *Horadrim*. 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:

***Horadrim* Definition Language Operations**

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

***Horadrim* Manipulation 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 primary key)

As the last member of *The Order*, you need to **obey the instructions** while designing the *Horadrim* system. Thus, the instructions shall ***include***, but are not limited to:

[1] https://www.diablowiki.net/Tale_of_the_Horadrim

- 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 alphanumeric characters inside the test cases.
- The hardware of *Horadrim* center and *Horadrim* instances will be built according to the blueprints, thus you do not need to consider the *Horadrim* physical storage controller to interact with the storage units.
- Note that type refers to a relation.

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:

- You will use B^+-Tree for indexing.
  - You will use primary keys of types for the search-keys in trees.
  - You should have separate B^+-Trees for each type, and these trees must be stored in file(s). So, when a type is created, its B^+-Tree will also be created and its structure should be stored.
  - With every *create record* and *delete record* operations, the corresponding tree should be updated.
  - You shouldn't generate B^+-Trees from scratch on the fly using records for DML operations. You should utilize the tree structures that you stored in files.
- 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 *Horadrim* 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.

3 Chapter Two: Implementation

In this phase, you are expected to implement the *Horadrim* software that you have designed in **Chapter One**. The implementation must support the ***Horadrim* Definition Language Operations** and the ***Horadrim* Manipulation Language Operations**.

4 Input/Output

Submissions will be graded **automatically** for various test cases. You will write your code in **Python3**. The input and output file names will be given as arguments to your executable program. The command that will be executed is as follows:

```
python3 2018400XXX/src/horadrimSoftware.py inputFile outputFile
```

Your directory must have the structure as shown below and the name of your main file must be `horadrimSoftware.py`.

```
2018400XXX.pdf
README.md
src
    horadrimSoftware.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:

Horadrim Definition Language Operations

Operation	Input Format	Output Format
Create	create type <type-name><number-of-fields><primary-key-order><field1-name><field1-type><field2-name>...	None
Delete	delete type <type-name>	None
List	list type	<type1-name> <type2-name> ...

Horadrim Manipulation 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><field1-value><field2-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

A sample 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 example test cases.

Sample Input File	Sample Output File
create type angel 3 1 name str alias str affiliation str	Tyrael AspectOfWisdom Horadrim
create type evil 4 1 name str type str alias str spell str	Itherael ArchangelOfFate HighHeavens
create record angel Tyrael ArchangelOfJustice HighHeavens	angel
create record angel Itherael ArchangelOfFate HighHeavens	evil
update record angel Tyrael AspectOfWisdom Horadrim	
list record angel	
list record evil	
list type	

Table 1: Sample Test Case - Input & Output

Log File
1623001321,create type angel 3 1 name str alias str affiliation str,success
1623001361,create type evil 4 1 name str type str alias str spell str,success
1623001371,create record angel Tyrael ArchangelOfJustice HighHeavens,success
1623001381,create record angel Itherael ArchangelOfFate HighHeavens,success
1623001391,update record angel Tyrael AspectOfWisdom Horadrim,success
1623001401,list record angel,success
1623001421,list record evil,failure
1623001431,list type,success

Table 2: Sample Test Case - Log File

5 Implementation Instructions

- You have to use B^+-Tree indexing to handle all *Horadrim* Manipulation Language Operations.
- Filter operation has a condition part, which can only filter according to the **primary key** and only less than $<$, greater than $>$, and equal $=$ operators can be used to create a condition.
- Any field of a type can be the **primary key** and it is defined with <primary-key-order>in the create type operation.
- Search, update and deletion of records shall always be done by **primary key**.
- Update operation updates every field other than **primary key**.
- When a type is deleted, all records of that type must be deleted.
- Types and records must be listed by ascending order of type names and primary keys, respectively. 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 and that's the expected behaviour.
- Test cases are not necessarily independent of each other. So, if the type angel 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.
- You can assume that only int and string field-types will be used.
- 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.
- *Horadrim* software must log all definition, and manipulation operations into a CSV file (namely horadrim-Log.csv) which consists of 3 columns, namely occurrence, operation, and status. 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 *Horadrim* software is either stopped or restarted.
  - The UNIX times in the sample test cases are set deliberately for you to understand the time flow between instructions and which instruction comes before which. In real life, a user may have different time intervals between instructions since they write the instruction string into the terminal, which also takes time. Your system may also print the same time output for some consecutive fast instructions, which is fine and there's no error in that. As long as you read the input file line by line sequentially, your output should be correct.
  - You can use `import time` and `int(time.time())` to generate UNIX times.
- 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. Some invalid operations that their status should be *failure* can be listed as follows:

- Creating a type with a name of an existing type (Suppose that type *angel* exists and you want to create the type *angel* again.)
- Creating a record with a primary key of an existing record (Suppose that type angel has a record with a primary key *Tyrael*, and you want to create another record of type *angel* with primary key *Tyrael*.)
- Deleting a type which does not exist in the database (Suppose that type *angel* does not exist and you want to delete type *angel*.)
- Deleting a record with a primary key which does not belong to any record of that type. (Suppose that type *angel* has no record with a primary key *Tyrael* and you want to delete the record with the primary key *Tyrael*.)

6 Report & Grading

You are expected to submit a report *written in LaTeX* that contains the sections below. LaTeX report template will be provided to you. Also, you have to provide a README file including your instructions to run ***Horadrim*** *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, file design, page design, page header, record design, record header, B^+-Tree design etc. with tables/diagrams/figures.
5. **Operations:** Explain your ***Horadrim*** **Definition Language Operations, and Manipulation Language Operations**. Refer to the **related files and functions** to improve the explanation.
6. **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 ***Horadrim***: 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. **Any other submission method is 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.**

8 Late Policy

You are allowed a total of 10 late days on the projects with no late penalties applied. You can use these 10 days as you wish. For example, you can submit the second project 4 days late, then the third project 6 days late. In that case, you will have to submit the fourth project on time. No late submissions will be accepted after you use these 10 extra days. If you change your team, the team's late days used so far will be the maximum number of late days used by any of the team members.