**DESIGN DOCUMENT**

Created by: Jian Xian, Marshall Chang, Tina Do

**GENERAL OVERVIEW OF OUR SYSTEM:**

Our program is written in python3 bsddb3 module for Berkeley database and sqlite3.

DIAGRAM

**A description of your algorithm for efficiently evaluating queries, in particular evaluating queries with multiple conditions and wild cards and range searches and an analysis of the efficiency of your algorithm**

**USER GUIDE:**

**Phase 1:** Phase1.py is called in the command line by typing: python3 Phase1.py

The user will be prompted for a file name. If a file is not found, it will continue to ask for a valid input. Once a valid input is received, the program will run for a short time depending on the length of the file. After completion, it will output the time it completed the process

**Phase 2:** Phase2.py is called in the command line by typing: python3 Phase2.py

The program will use the four text files created from Phase1 and sort, remove duplicates, and reformat to be used by db\_load shell command to generate four index files.

Using the bash shell and script way:

In the linux command line or bash window. Navigate to the folder containing the files created from Phase 1. Type “./Phase2\_sort.txt” in the command line. It will sort the files and remove any duplicate lines. The files remain the same name

**Phase 3:**

**DESIGN OF SOFTWARE:**

Phase1: Script that handles reformatting of xml files. There is a method for each file to parse xml style.

Usage: python3 Phase1.py  
 >>[enter xml file name]

Phase2: Script that automatically creates index files from the files in Phase1.

-Loads Phase 1 Files (subprocess module)

-Sorts the files with Linux Sort (subprocess module)

-Parses the results from Phase 1 into index files’ key, data line by line format.

-Creates index files (with bsddb3 and subprocess module)

//Use inside the same folder that Phase1 was run in.

Usage: python3 Phase2.py

Phase3: Command line interface which searches the index files based on users input.

-Takes user’s command line input for search query

-Parses the user’s query with regular expression and string methods

-Searches index files for each user query term, and creates a list for all found under that term

-Intersects all the lists to finally return results that match all search terms

Usage: python3 Phase3.py

//follow instructions of program

**OUR TESTING STRATEGY:**

1. For basic functions testing, we test our program as we go, ensuring all examples on the assignment page work.

2. Also code review is done to make sure that it can handle all cases.

3. Additional error checking is done to make sure the program does not break from simple user mistakes.

**GROUP WORK BREAK-DOWN AND DETAILS**

**Group Meetings:**

**11/09/19 – 1h**

**11/25/19 – 1h**

**Individual Development Time- Includes Design, Development, and Testing**

**Marshal:**

**P2:**

Python script to automate Linux commands using subprocess module **(0.25h)**

File parsing algorithm (Based on Perl Script from eclass) **(0.25h)**

**P3:**

Parse user input algorithm and error checking **(1h)**

Range search (Re-purposed from lab slides Range Search) **(0.25h)**

Intersection algorithm **(0.5h)**

**Testing:**

Code review for all parts **(1h)**

Testing phase 3 for all examples in specifications and variants **(0.5h)**

**Misc:**

Helped other group members to debug code and give advice **(2h)**

**Jianxiang:**

**Phase 2: 1h**

**Phase 3: 3h**

**Tina:**

**P1: ~1.5hrs**

Created python script to take XML files and produce four text files

**P2: ~2hrs**

Researched and added Linux commands sort and Berkeley commands db\_load and db\_dump

**P3: ~4hrs**

Created template and user interface

Get Records Brief algorithm that retrieved brief records from the database

Error checking on get records brief