CMPUT 291

Mini-Project 2

Group ID: g13

Group Member’s names (CCID):

Justin Daza (justinjo)

Siddhant Khanna (skhanna1)

Klark Bliss (klark)

We did not collaborate with anyone

Overview:

* Our program is split into three source code files named phase1\_script.py, phase2\_script.py, and phase3.py.
* phase1\_script.py reads records in records from standard input and produces 3 files described below
  + terms.txt: This file includes the terms extracted from the title, author, journal, publisher and book title field of all records.
  + years.txt: This file includes one line for each record in the form y:K where y is the year of the publication and K is the record key.
  + recs.txt: This file includes one line for each record in the form K:rec where K is the record key and rec is the fill record in xml.
* phase2\_script.py sorts the files built by phase1\_script.py using the Linux sort command; it pass the right option to the sort command to keep only the unique rows. We store the sorted files into three separate files in the form s\_”filename” where file name is the name of on of the phase 1 created files. This file also creates three indexes (1) a hash index on recs.txt with record key as keys and the full record as data, (2) a B+-tree index on terms.txt with terms as keys and record key as data, (3) a B+-tree index on years.txt with years as keys and record key as data. You should note that the keys in all three cases are the character strings before colon ':' and the data is everything that comes after the colon.
* phase3\_script.py Given the index files re.idx, te.idx, and ye.idx created in Phase 2 respectively on record keys, terms and years, the program processes queries as follows. By default, the output of each query is the key of all matching records. The user should be able to change the output format to full record by typing "output=full" and back to keys only using "output=key". Here are some examples of queries:

1. title:parallel
2. author:schindler
3. year:2000
4. other:pvldb
5. database
6. year>2006 year<2010
7. title:paralel title:sorting
8. title:"main memory data platform"
9. year:2000 other:pvldb database

* The first query returns all records that have the term parallel in their title field; the second query returns all records that have the term schindler in their author field. The third query returns all records with year 2000, and the fourth query returns all records that have the term pvldb in other text fields (those fields include journal, book title and publisher). The fifth query returns all records that have the term in title, author or other text fields. The sixth query returns all records with a year larger than 2006 and less than 2010. The seventh query returns the records that have both parallel and sorting in their title. The eight query returns the records that have the given text as a substring in their title field. Finally, the last query returns the records that match all three conditions.
* More formally, each query defines some conditions that must be satisfied by title, author, year and other text fields of the matching records. A condition can be either an equality match or a range match. All matches are case-insensitive, meaning the queries "database", "Database", "dataBASE" would retrieve the same results; for the same reason the extracted terms in previous phases are all stored in lowercase. Matches on year can be exact (as in query 3) or range searches (as in query 6). Matches on title, author and other text fields can be based on one term (as in queries 1,2,4,5), multiple terms (as in query 7) or a phrase match (as in query 8). A query can have multiple conditions (as in query 9) in which case the result must match all those conditions. [Here](https://eclass.srv.ualberta.ca/pluginfile.php/3632962/mod_page/content/27/ql-grammar.txt) is a grammar for the queries.

Detailed Design:

* Phase 1 imports the os and re libraries. It has everything managed by the main function wherein it queries the user for the initial input and then creates the three files outlined above. Through the use of loops and the .findall() function the files are populated and closed.
* Phase 2 imports the os and subprocess libraries. The main function of phase2\_script.py checks if terms.txt, years.txt, recs.txt exist and if they do it calls createSortedfile(filename): to use the linux sort command on them. Then it creates the index file for each of the files including s\_recs.txt, s\_terms.txt, s\_years.txt using a function for each sorted file:
  + def CreateRecsIndex(sorted\_filename): creates hash index for s\_recs.txt
  + def CreateTermsIndex(sorted\_filename): creates B+ tree index for s\_terms.txt
  + def CreateYearsIndex(sorted\_filename): creates B+ tree index for s\_year.txt
* Phase 3 imports db from bsddb3, re, copy and shlex libraries. It has a basic querying interface that asks the user for input where they can phrase their input in the forms:
  + Key:value
  + Key:“Value Value Value”
  + Key<Value (numeric)
  + Key>Value (numeric)
  + Value
  + (Or any combination of the above separated by whitespaces

The main function creates an input loop for the user, checking for an “=” to see if the user wants to switch out from keys (default) to full or vise versa. The input query is sent to the function ProcessQuery(query) where all three databases are loaded and opened with all three index files. Then in input is split into a 2D array of key, value pair lists of size 2 each. (ie. [[key, value], [key, value], …, [key, value]]). Then based on the key the function passes each [key, value] pair into its respective function.

* + BlanketSearch: If the user enters a value without a key this function will be called and use functions TitleSearch, AuthorSearch, and OtherSearch to search through all three databases to see if they contain the search value.
  + YearSearch: If the user enters the key as “year” followed by a colon and number then this function will retrieve all records that match the given year
  + AuthorSearch: Retrieves all records with the given author listed. Supports quotation search
  + TitleSearch: Retrieves all records with the given title listed. Supports quotation search
  + RangeYearSearch: If the user enters the key as “year” followed by an inequality symbol such as (< or, >) and a number then this function will retrieve all records that are greater than or less than the given year. Supports multiple range inputs.
  + OtherSearch: This function will retrieve records from any value not listed above

Testing Strategy:

* We created an input file generator that would create a file of one hundred thousand lines which we could test on.
* Also, we tested our functions through the use of print statements and using agile software development strategies where we programmed in pairs and in threes.

Group Work Break Down:

* Justin Daza: Phase 1, Phase 3 (Query 1,2,3,4,5,6,7)
  + Hours Spent: 18
* Siddhant Khanna: Phase 2, Phase 3 (Query 1,2,3,4,5,8,9), bug fixing, testing
  + Hours Spent: 35
* Klark Bliss: Phase 3 (Query 1,2,3,4,5,6,7,8,9), Input File Generator, bug fixing, testing
  + Hours Spent: 17