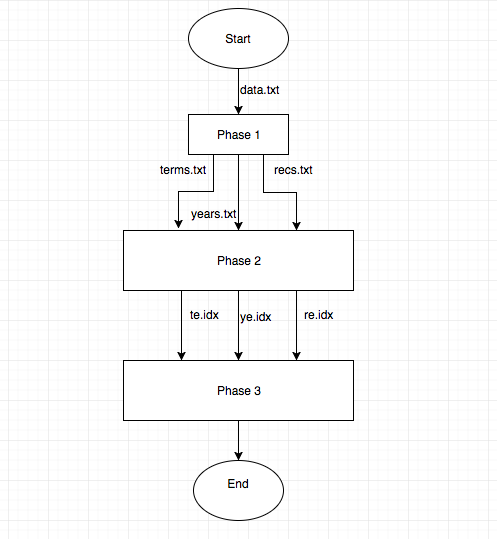
**Contributors:**

Aman Anand

Zhihao Zhang

Adit Hasan

**General Overview:**

1. The program starts with running Phase 1 and creating 3 files
2. Then Bash Script is run to sort the data and create 3 indices on the 3 files
3. The Phase 3 is run to evaluate the queries according to the input provided from command line or reading from file and printing it according to the output format provided by the user which is by default “output=key”.

Phase 1:

1. Parsing the file by xml tag
2. Extract different information from the xml files
3. Storing different information into 3 different .txt files : term.txt , year.txt, recs.txt

In Phase 2:

Bash Script file which:

1. Sorts the 3 files
2. Parse it by break.pl according to input which db\_load accepts
3. Making 3 indices by db\_load:  te.idx, ye.idx, re.idx

In Phase 3:

1. Parsing the input from user
2. Passing it to the right function to evaluate the query
3. Printing the results based on the output format

**User Guide**

Instructions for running the code :

Move to the directory where all files(parser\_xml.py , Script, phase\_3.py, break.pl, print\_full\_record.py) exist

Run the following commands in order from the terminal:

1. python3 parser\_xml.py
2. Bash Script
3. python3 phase\_3.py

**Algorithms for evaluating queries**

1. Years query :

Uses 3 sets() :

1. Flag\_greater\_than : takes all the greater than years from the input

Flag\_less\_than : takes all the less than years from the input

Flag\_eq : takes all the equal years from the input

     2.   Picks the min of the set() flag\_greater\_than

Picks the max of the set() flag\_less\_than

     3.   Checks if the flag\_eq lies between smin and max

If no then no output

If yes then print the results for the equal year

If len equal flag is zero then returns the result bw min and max

By default min is inf and max is -inf

       2) Phrases :

Checks if the input is title/author/other + : + in double quotes

If yes then splits each word in the phrase and checks in the te.idx

If all words matches then checks in re.idx if the phrase is in order

       3) Terms :

Checks if the input is just a word or title/author/other + : + word

If yes then check the word in te.idx and returns the result

Testing Strategy

Phase 1:

Testing and Bug fixed

First of all, I was using 10 records for Phase one implementation.After finishing the code, I parsed 1000 records to compare 3 files with the result provided in the eclass.

In Phase 1, The string less than 2 didn’t removed at very beginning. I was aware by getting wrong answer in the query and eventually fixed it.

Phase 2:

Testing and bugs fixed:

Command db\_load -p file was used to test the correctness of the indices

Caught the mistake where data was not sorted correctly as the parameters were not correct while running the sort commands

Phase 3:

The same testing protocols were used for files with 10, 1000 and 20,000 records to make sure the functionality was consistent across files of different sizes.

Queries were tested out with focus on edge cases. Queries that are expected to not output any records were tested for accuracy and those that should print out all available records were used to test speed.

Testing and bugs fixed:

Query 8 was wrong while testing as the phrase was not in order in the title but still came as an output. It was fixed by checking the re.idx and parsing the record and getting the exact title and then matching the order with the input phrase.

Group Work Break-down Strategy

Regular twice-a-week meetings were used to coordinate efforts and update the team of each member’s progress.

The work done by each member is as follows:

Adit Hasan (~ 2 days)

1.Searching the records file for full records when output=full

2.Parsing and printing out the xml records

Aman Anand (~ 2 days)

1.Phase 2 & Phase 3 Implementation except printing when output=full

2.Zhihao Zhang helped me in Query 8.

Zhihao Zhang (~ 2 days)

1.Implemented Phase one  and all parsing related method

2.Helping Aman Anand into Query 8