# Mini Project 2 Design Document

## By Joey-Michael Fallone and Tony Qian

# Overview:

This program constructs and queries a database of Amazon product reviews using the Berkeley Database package in its linux and python3 modules. Phase one of the program assigns a review number to each review and then parses the raw data into four files:

rterms.txt – Stores alphanumeric terms from the review text as keys to the review number

pterms.txt – Stores alphanumeric terms from the product name as keys to the review number

scores.txt – Stores review rating as a key to the review number

reviews.txt – Stores the review number as a key to all raw data from the review, including the

original review text

Phase 2 of the program sorted the phase one files alphanumerically and then used the db\_load linux command to pipeline the data into Berkeley db files. The ‘./dump.sh’ script can also be used to visualize the data in the idx (chosen extension for Berkeley db files) files. It’s also important to note that all of these files were made using a BTREE index scheme, with the exception of rw.idx, which used a hash index to relate the review number to the rest of the review data.

Phase 3 of the program interprets queries from the user and searches for the corresponding review numbers. It returns the number of reviews found matching the queries with the first review, and prompts the user to continue seeing the rest of the reviews on at a time.

# User Guide:

Phase 1:

To build the text files run:

cd p1 // move to the phase 1 dir

./make.sh // runs script to delete legacy files and create new ones

cd .. // move back to main directory

Phase 2:

To sort and build the index files run:

cd p2 // move to phase 2 dir

./build.sh // runs script

cd .. // move back to main dir

Phase 3:

To run main program

cd p3 // move into phase 3 dir

python3 main.py // run program

cd .. // move back to main dir

IF ANY BASH SCRIPTS GIVE PERMISSION ERRORS RUN THE FOLLOWING:

chmod + x name\_of\_problematic\_script.sh

# Algorithm Description

The initial evaluation of the queries involved some significant parsing. Certain characters were identified and spaces were placed around them if they had not been yet. The string was then split into a list of strings for individual analysis. Then a function was used to ensure that the first query in the string was either a reference to “rscore” followed by a greater/less than sign and a floating point number, or a search term such as “camera” or “r:great.” If this was not the case, the function would shift the string over and rearrange the queries so that it was.

Each singular query, such as rscore or search term queries, was evaluated alone. These results were then converted into sets. This served two purposes, removed any duplicates, which could be an issue in “%” searches, example “cam%,” and allowed the use of set operations. The results were joined together and the result was converted back into a list and sorted in alphanumeric order. When a non-singular query was encounter, for example “rdate > 2007/06/01,” at least one singular query was run first and the results from it were searched for their compliance with the non-singular condition. The list of results that remained complied with both queries and was now independent in that it was free to be used in a set join with other independent results, to be used as a result to be searched for compliance with another non-singular query, or to be returned as the solution.

# Testing Strategy:

We did testing for all phases of the project including reviewing the text files for Phase 1, the idx files for Phase 2 and the query results in Phase 3. We tested to see if our output files in Phase 1 were correct by having a partner look for differences between our .txt files and the ones obtained from drafiei/291/pub and having the other find ways to fix it. For example if some lines in our pterms.txt had weird characters then other partner would go into our python script and add a way to remove them until our .txt files matched theirs. We also had to see if our .idx files were created properly and if db\_load and sort were working well. We did db\_dump’s on our .idx files to check for proper sorting and duplicate handling and to see if the formatting in them looked correct. We also checked that the info inside was consistent with our .txt files and queries results. For Phase 3 we ran various queries that we brainstormed together. One partner would run these queries while the other confirmed the results from print by searching for these results in the physical review.txt file to check if anything was off and if the results were consistent and all data was being returned and displayed properly. We also tried to approximate if the correct number of results are returned by searching for our query results in the Phase 1 files and seeing if the numbers were similar. For example if searching great in rterms.txt was close to the number of results of r:great.

# Group Work Break-down Strategy

Our strategy for breaking down the work for this project was to first work together to finish Phase 1 and 2, creating the .txt files and making the .idx files. We both performed some of the work for these phases such as Tony did the method for replacing non-alphanumeric characters in the input when creating pterms and rterms, while Joey did the method for formatting the output files themselves and reading in the input. Phase 2 Tony did the bash scripts for sorting the files and running db\_load while joey created a python script to add newlines between each key and data pair to format the files for db\_load. After completing parts 1 and 2 we split up part 3 evenly with each of us discussing what methods what be needed and then deciding who would create which ones. We did the testing and design doc together.

**Tasks by member (allocated time) [progress made]:**

**Tony [Spent ~10 hours of individual work]:**

Implement replacing double quotes and backslashes for Phase 1. (0.5 hour) [100%]

Implement removing non-alpha numeric characters for Phase 1. (0.5 hour) [100%]

Bash script for sorting Phase 1 files and running db\_load on the files after formatting (2 hour) [100%]

Create bash script for running all phases and clean all script (1 hour) [100%]

Create print reviews function for Phase3 (2.5 hours) [100%]

Create functions for querying in Phase 3 (3 hours) [100%]

Assisting in creating search functions for Phase 3 (2 hours) [100%]

**Joey [Spent ~ 12 hours to work individually]:**

Implement formatting and grabbing the right data from input for Phase 1. (1 hour) [100%]

Python script to insert newlines between key and data pairs for Phase 2. (0.5 hour) [100%]

Helping do db\_load script and testing idx files with db\_dump (1 hour) [100%]

Load db\_objs, create prompt and set up main function for Phase 3 (2 hours) [100%]

Create functions for querying in Phase 3 (3 hours) [100%]

Assisting in Phase 3 query functions (2 hour) [100%]

Create search functions for Phase 3 (4 hours) [100%]

**Cooperation [Spent ~ 17 hours to work together]:**

Implement Phase 1. (2 hours) [100%]

Implement Phase 2. (3 hours) [100%]

Implement Phase 3. (8 hours) [100%]

Testing (2 hours) [100%]

Work on design doc (2 hours) [100%]