CMPUT 291 Mini Project 02

Will Fenton
Alexis Seniuk
Abdulazeez Ojetola

November 26, 2018

***** COLLABORATION:**

- Our group collaborated on the source code through github
- We communicated (and planned meetings) through WhatsApp
    - Our group met roughly once per week to ensure everyone was on track with the project

System Design Document

1 General System Overview

Description

- We built an information retrieval system that implements the properties within the Berkeley DB library
- The program is a 3 phase process outlined further below:

Phase 1: Prepare Data Files: creates 4 txt files from raw xml inputs	1. Terms.txt: 2. Pdates.txt: 3. Prices.txt: 4. ads.txt:
Phase 2: Build Indices: sort files in Phase 1 so they can be piped to phase3 loading program	(1) a hash index on ads.txt with ad id as key and the full ad record as data, (2) a B+-tree index on terms.txt with term as key and ad id as data, (3) a B+-tree index on pdates.txt with date as key and ad id, category and location as data, (4) a B+-tree index on prices.txt with price as key and ad id, category and location as data
Phase 3: Data Retrieval: Queries functions built to handle the possible input queries outlined by the grammar file from the project specification	→ dateQuery → priceQuery → locationQuery --> categoryQuery → termQuery

- Our source code is written in python and uses the Berkeley DB library for building and accessing the information on the files and indices
- The system operates through a simple command line interface.

Install & Run

- Run the program through the following command line prompts:
- **FIRST** ensure credentials:
  - > chmod +x phase1.py
  - > chmod +x break.pl
  - > chmod +x script.sh
- **SECOND** run the script file that handles phase 1 and phase 2
  - > ./script.sh [file-name for xml input file]
- **THIRD** run the main program
  - > python3 phase3.py

User Guide: System Breakdown

- **Main Menu:** Phase3 is the main implementation of the program; the user is taken to the main menu to begin
  - The main menu has 3 options that the member can select from
  - At any time the user can type exit to quit OR back/enter to go back to the authentication menu

Main Menu Options	Functionality
1. Terminal (enter queries / change output mode)	● At this menu option, the member can query the databases ● The main queries the user can search for are: * FURTHER query handling can be found in the second menu option 'View Query Grammar' ● The user has the ability of changing the output mode from BRIEF <-> FULL

2. View Query Grammar	• The query grammar specification outlines can be viewed through the second menu option
3. Exit	• The third menu option exits the member from the program * NOTE * Nothing is committed nor changed in the databases; the user does not have write privileges

---

2 Software Design &
3 Testing

- We tested our program functionality throughout the whole construction. The project was designed to be composed of a series of functions, classes, modules and scripts each of which had a unique role, hence allowing testing to run smoothly. In addition, when combining our portions of the system, our group ensured that variable names, formatting and the structure of the functions were uniform across the system. This provided consistent code distribution and permitted ease to sample testing during the system construction as well as fixing merge bugs towards the end of the project.
- Included in the testing, we tested for error handling, variable constraints when passed, input constraints, database commits and changes

Phase	**Function Description and Purpose**		
	Functions		**Purpose/Role**
Phase 1	[1] terms_function(terms_file, ad_data) [2] pdates_function(pdates_file, ad_data) [3] prices_function(prices_file, ad_data) [4] def ads_function(ads_file, ad_data) [5] def process_ad(raw_ad)		[1] Extracts terms from the ad's title and description and writes them to terms.txt [2] Writes posting date information to pdates.txt [3] Writes price information tp prices.txt [4] Writes ad information to ads.txt [5] Extract ad data from the raw record string; Returns dictionary containing data about the ad
Phase 2	permissions.sh break.pl clean.sh script.sh		[1] grant permissions to files [2] clean files [3] parse files [4] sort files [5] create .idx files [6] dump .idx files
Phase 3	**1. Class QueryParser**	[*]__init__(self) [1] close_databases(self) [2] set_query(self, query [3] parse(self) [4] price_query(self, query) [5] date_query(self, query) [6] location_query(self, query) [7] category_query(self, query) [8] term_query(self, query) [9] print_query_conditions(self)	[*] build databases and regular expressions [1] close databases [2] build query bins [3] parses query and sorts into bins [4] price query handler (<>=<=>=) [5] date query handler (<>=<=>=) [6] location query handler (exact match) [7] category query handler (exact match) [8] term query handler (exact and partial(%) match) [9] print/display queries
	2. Class Interface	__init__(self) [1] clear_screen(self) [2] main_menu(self) [3] terminal(self) [4] grammar(self)	[*] [1] clear screen [2] main menu (options 1,2,3) [3] handles raw input for queries [4] displays the grammar assignment specifications change mode from BRIEF <-> FULL

Query Optimization

- Query optimizations were considered for considered for handling the terminal input
- Examples handled are as follows:

Date Queries	➔ If 2 dates are entered, the greater one is queried
Term Queries	➔ If for example term% and term are queried, both queries will be executed as they are handled as 2 independents
Price	➔ The greater or lesser value is handled for the price ➔ For example: If price > 10 and price > 15 then only the matches greater than 15 will be returned as there would be none between 10 and 15 for the intersections
Category	➔ Multiple category queries cannot be entered
Location	➔ Multiple location queries cannot be entered

- Our program was also designed to only access the required indexes when duplicate keys can be found across the 4 different databases
    - Further:
        - Date Queries → the cursor is set to iterate through the date index
        - Term Queries → the cursor iterates through the terms index
        - Price Queries → the cursor iterates through the prices index
        - Category Queries → the cursor iterates through the prices index
        - Locations Queries → the cursor iterates through the prices index

---

4 Bugs & Improvements

- If more time was permitted, we would have liked to implement the following functionalities:
    - More consistent formatting and error handling throughout the entire program
    - Decrease the runtime of the program by grouping functions and removing redundancies
- Implement user injection or database manipulation restrictions
- Include broader input query handling for an expanse of different values (could be a string, number or NULL) to fit the assignment specifications

---

5 Group Work

Breakdown Strategy

	Abdulazeez	**Alexis**	**Will**
Time Spent	~20 hours	~20 hours	~20 hours
Contribution	● Phase 3 queries: 1. Date query 2. Price query 3. Combine intersection of adIDs	● Design Document ● Phase 2: build index ● Phase 3 queries: 1. Location query 2. Category query 3. Term and Term% query	● Phase 1: prepare data files ● Phase 2: scripts for building indexes and ensure proper formatting ● Query Optimization ● Main Menu/Interface ● Class Structure planning ● Testing