COMP 4321
Group 4
Jia Lu, Sam Baltrus, Adam Feuer

Phase 1 RocksDB Database Scheme

RocksDB name	**Mapping**
wordToWordID	word <-> wordID ● word: `String` ● wordID: `int` ● Uses the Java hashCode() function to hash a keyword to a unique wordID ● Contains all words found in all documents indexed by the crawler Design Explanation/notes: ● Before the word is inserted into the database, it is first stemmed using Porter's Algorithm. The word is not the original keyword that appears in the document, but instead the stemmed version of the word ● During implementation for queries, we plan to stem all words using the same algorithm to ensure consistency in retrieving the correct wordID from the same words.
urlToPageID	url <-> pageID ● Url: `String` ● pageID: `int` ● Uses the Java hashCode() function to hash a url to a unique pageID ● Contains all urls found in all documents indexed by the crawler
pageInfo	pageID <-> Page ● pageID: `int` ● Page: a `Page` object ○ The `Page` class contains the following fields ○ page title: `String` ○ url: `String` ○ Last modified date: `String` ○ Page size: `int` ○ Child links: `ArrayList<String>` Design Explanation/notes: ● The Page class is used to store meta information about each page, so we added this RocksDB to make it easy to access this information directly from the pageID.

invertedIndex	wordID -> pageList(a list of pages that the word appears in) ● wordID: `int` ● pageList: `HashSet<Integer>`
forwardIndex	pageID ->wordList(wordIDs of words that appear in the webpage, and the position of the words in that page) ● pageID: `int` ● wordMap: `HashMap<Integer, ArrayList<Integer>>` ○ wordMap uses the wordID as the key. The value that correspond to the key is an Arraylist that contains the positions of that wordID for that particular pageID Design Explanation/notes: ● We stored the positions of the word along with the word itself. By retrieving the words in addition to their positions, it is easier to support searching by phrases