COMP4321 Group 3 Project Phase 1 – JDBM design

The following 4 tables highlight the design of how data from crawled pages are stored. Underlined attributes are keys of each relation in this document. In implementation, the relation key is used as the key for JDBM trees (all of which uses `HTree`), and values are implemented as structs. For example, (`pageID, pageLastMdf`) of `forwardPageIndex` will be wrapped in a struct called `forwardPageIndexData` that will be retrieved, mutated, and then stored back into `fowardPageIndex`. We believe this design makes the code easier to understand.

`forwardPageIndex`

pageURL	pageID	pageLastMdf
testpage	p0	16 May 2023
ust_cse	p1	16 May 2023
…	…	…

As the spider does BFS crawl new pages will be discovered. Each new page will be given a unique pageID (as UUID string) and its last modified date is also recorded. The page will not be recrawled (handle cyclic links) if the URL already has pageID and modified date has not been changed.

`backwardPageIndex`

pageID	pageURL	pageTitle	pageWords	childURLs
p0	testpage	Test Page	{w0: {ps_0, ps_1}, …}	{url0, url1, …}
p1	ust_cse	CSE Department of HKUST	…	{…}
…	…	…	…	…

`pageWords` is a hashmap stored for each page, using `wordID` (see below) as key and a vector as value. The value vector stores for each word all the positions it appears in on this page. `childURLs` is a String vector that stores all the child links on the page. Redundant `pageURL` to improve performance.

`fowardWordIndex`

stemmed	wordID
test	w0
page	w1
…	…

Used to add words into the record; once a new word is discovered an ID will be created for it. Words will be stemmed using Porter algorithm first. Stop words and words completely pruned by Porter will be ignored. Both title and body keywords will be given the same set of wordIDs.

`backwardWordIndex`

wordID	stemmed	wordPages
0	test	{p0: {pos_0, …}, …}
1	page	…
…	…	…

The inverted index for keywords. `wordPages` is a hashmap using pageID as key and word positions vector as value. Redundant stemmed word also for performance improvements.

pageTitleIndex

wordId	titleWordPages
0	{p0: {pos_0, …}, …}
1	…
…	…

Like the inverted word index above just with words from page titles. Fulfills the requirement "all stems extracted from the page title are inserted into another inverted file". May potentially support title search.