COMP 4321 Search Engines for Web and Enterprise Data
Project Database Design Document

LI, Kwan To
ktliac@connect.ust.hk
LUK, Sui Hei
shluk@connect.ust.hk
WONG, Yan Yuet
yywongaw@connect.ust.hk

March 20, 2021

1 Overall Design

In the current phase, we had three tables to support the program which are Forward Index, Document ID Lookup Table and Inverted Index. The Forward Index and Backward Index are the core of the search engine - the Forward Index maps document IDs to document record objects which contain details of the document while the Backward Index maps the keywords to posting lists which consist of the term frequencies of the words. Meanwhile, the Document ID Lookup Table is an auxiliary table that maps the URLs crawled to respective document IDs.

To update the index after crawling, we need to first lookup the URL in the Document ID Lookup Table and assign a new Document ID to the URL if it does not exist. Then all the details of the document will be further updated to the Forward Index along with the assigned Document ID. To update this record to the Inverted Index, we would lookup each keyword specified in the document basic details and further update the posting list if needed. For a new keyword, we would insert a new key-value pair by having the keyword pointing to DocID and the frequency of the word in the document. If the keyword is found, we would simply add the DocID and frequent to the posting lost.

2 Forward Index

2.1 Schema

	Field	Type	Description
Key	Document ID	`Integer`	The document ID of the document to be looked up.
Value	Document Details	`DocumentRecord`	The document record details of the respective document.

2.2 Design

The Forward Index table is used for storing the basic details of each document represented by their Document ID. The details to be stored include the page title, URL, last modification date, size of the page, keywords and their respective frequencies and the child links.

Our design ensures the details of the document can be retrieved efficiently given the document ID. Here, we store the document record as a single object instead of using separate tables -

one would be able to get all the details instantly in one single table access. This is especially important as the search engine is required to list out document details frequently during the search process. It would be too costly having to search through different tables looking up the results.

3 Inverted Index

3.1 Schema

	Field	Type	Description
Key	Keyword	`String`	The keyword to be looked up.
Value	Posting List	`Map<Integer, Integer>`	The map that stores the term frequency of the respective keyword of documents.

3.2 Design

The Inverted Index table stores the document IDs that contain the keyword and their respective frequency. This is to cater the need for building a vector-space model later used for ranking the search results. In such a model, the similarity between a query and a document is to be calculated based on TF-IDF. That is, we have to support the efficient lookup of term frequency and document frequency given a set of query keywords.

When user input search queries, we can quickly lookup the inverted index table to get all the document IDs of the documents containing the keyword and also the frequency of the keywords in the document. Instead of having to search through the values in the Forward Index to find all the documents that contain the keywords specified in the query, the inverted index table speeds up the lookup process, thus giving better performance.

We will consider adding other properties such as keyword position to support phrase search in the later phase of the project.

4 Document ID Lookup Table

4.1 Schema

	Field	Type	Description
Key	URL	`String`	The URL to be looked up.
Value	Document ID	`Integer`	The document ID of the document pointing to the respective URL.

4.2 Design

The Document ID Lookup table stores the Document ID of a certain URL. By having an integer Document ID instead of a string URL representing each document, it reduces the storage and computational costin storing document-related information in other tables and fosters easier and faster searching.