Search Engine Project Phase 1 - Database Design

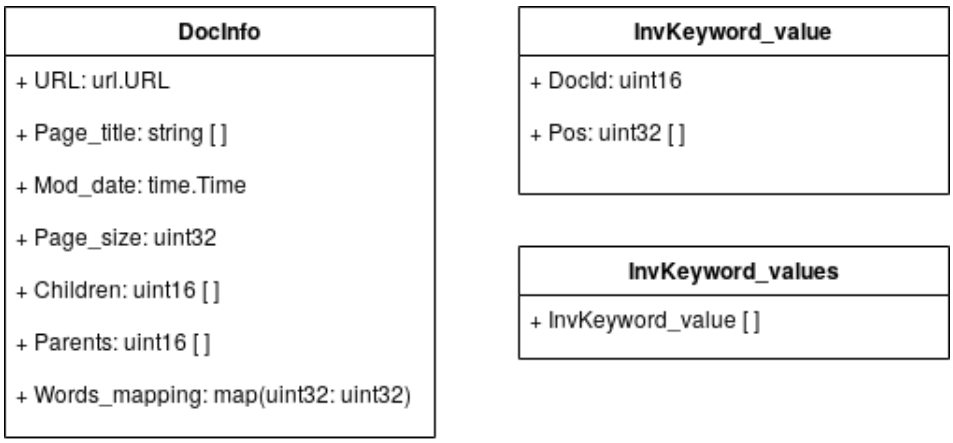
Group 12

May 2, 2019

1 Overview

We use BadgerDB[1], a KV database written purely in Go that uses LSM tree. The database mainly consists of two table types: inverted and forward. In addition, several data types are defined to support the data stored in the tables

2 Data Type



uint16 is used as the document id type because of the number of pages scraped in this phase is limited to 30. As uint16 has the maximum value of 65.535, uint16 enables the system to store and index 65.535 webpages while still not consuming much memory when stored in the database.

In addition, uint32 is chosen both for word id type and word frequency due to maximum number which reaches 4.2 billion. As number of unique words or frequency of words could easily reaches the maximum value of uint16, uint32 was chosen instead.

2.1 DocInfo

Document refer to a single webpage that is scraped from the crawler. DocInfo is a class containing the URL, page title, last modified date, page size, list of id of the children as well as the parents, and the mapping on the frequency of the word.

[1]https://github.com/dgraph-io/badger

2.2 InvKeyword_value

The InvKeyword_value and InvKeyword_values are defined to support the value of the inverted table. InvKeyword_value contains the document id and list of the position the word occurring in the document. InvKeyword_values is only the list of the InvKeyword_value help getting and putting the data into inverted tables

3 Inverted Tables

Inverted Tables (body)	
K	**wordId (uint16)**
V	InvKeyword_values

Inverted Tables (title)	
K	**wordId (uint16)**
V	InvKeyword_values

Two inverted tables were used: one for inverted tables for storing the mapping from word id to list of document id with their respective position on the body, and one for storing from the page header. The key of the Inverted table must be the word id, having type uint32, and the value is InvKeyword_values.

The inverted tables mapped word id with the document (corresponding to document id) that contains the word (related from word id, as well as the position of the word occurring in the tables. The position is stored in order to support the phrase search (i.e. "hong kong"). The value of each inverted table is a list or array of InvKeyword_value, each containing the document id (having uint16 type) and list of position the word corresponding to the word id occurs (type: uint32).

4 Forward Tables

Forward Table word	
K	**word (string)**
V	word ID (uint32)

Forward Table word ID	
K	**word ID (uint32)**
V	word (string)

Forward Table URL	
K	**URL (url.URL)**
V	doc ID (uint16)

Forward Table doc ID	
K	**doc ID (uint16)**
V	docInfo

4.1 word <=> wordId mappings

For mappings between word (type string) to its id (type: uint32), two different tables are used: one to map word to its id (key: word and value: word id) and the other one to map word id to word (key: word id and value: word). This is done to make sure that the search engine were able to get the index of the word given the word, or on the contrary, faster.

4.2 URL <=> docId mappings

Each webpage is indexed through document id. 2 tables are used in order to store the index of the webpage, with the same principle as the mappings between word and wordId. As one table is used to map URL (having type url.URL) to document id (type uint32), the other one is made to map document id to document info (type DocInfo).

Document info is a struct defined to store the different statistics and information about each webpage (refer to section 2.1). From this table, the search engine is able to get the URL of through DocInfo given the document id.

5 Helper Tables

Table next ID	
K	**next ID (string)**
V	biggest value (uint32)

5.1 biggest ID values

To ensure all the threads are in sync, this table stores the next wordId and docId that a new term or page can hold, respectively. Thus, this table has indexType (type: string), that is either "nextWordID" or "nextDocID", as key and the biggest indexValue (type: uint32) as value.