COMP4321 Spring 2017

Project Phase 1

Database Design Document

littleGoogle

Author:

Arslan Arshad (20216202, mohammad)
Mang Tik Chiu (20276758, mtchiu)
Yao-Chieh Hu (20216239, yhuag)
Yu-Ning Huang (20213421, yhuangbl)

Overview

The phase 1 of this project supports a database that constitutes of 4 tables, aiming to facilitate both the effectiveness and efficiency of the search engine design. The tables are listed below:

- Inverted Index Table
- Forward Index Table
- Link Index Table
- MapTable (Word & URL)

More details about the tables are to be discussed in the following of this document.

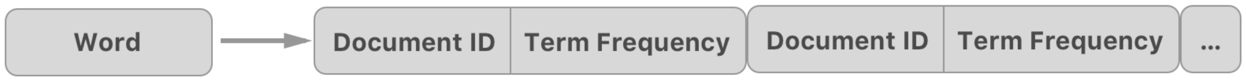
Database Design

I. Inverted Index Table

1. Introduction

Given a specific term/word, this table is to store a list of documents and the corresponding term frequency. It speeds up the relevant documents searching when providing a related keyword.

2. Diagram



3. Implementation

There are two internal functions supported for entry updating and entry retrieving:

1. **updateEntry(word, doc_id, freq)**

    Aims:

    This function is used in Crawler.java in **updateWordIndex()**, who updates the Inverted Index table when all the documents is crawled and stored.

    Parameters:

    word - String type; The term to be inserted as the key.
    doc_id - String type; The ID of a document that contains this term.
    freq - String Type; The frequency of appearance of this term in this document.

2. **getEntry(word, doc_id)**

    Aims:

    It retrieves the term frequency back as the result, given a specified word and document ID.

    Parameters:

    word - String type; The term to be searched as the key.
    doc_id - String type; The ID of document in the posting list of this word

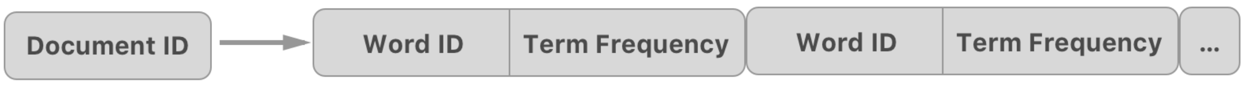
Return:
freq - String type; The frequency of appearance of this term in this document.

II. Forward Index Table

1. Introduction

Given a document ID, this table is to store a list of word IDs and its corresponding term frequency. It provides some vital information such as the length of a document and the total size of document stored to be searched.

2. Diagram



3. Implementation

There are two internal functions supported for entry updating and entry retrieving:

1. **updateEntry(doc_id, word_id, freq)**

   Aims:
   This function is used in Crawler.java in **updateWordIndex()**, who updates the Forward Index table when all the documents is crawled and stored.

   Parameters:
   doc_id - String type; The ID of the document to be inserted as the key.
   word_id - String type; The ID of the term containing in this document.
   freq - String Type; The frequency of appearance of this term in this document.

2. **getEntry(word, doc_id)**

   Aims:
   It retrieves the term frequency back as the result, given a specified word and document ID.

   Parameters:
   word - String type; The term in this document.
   doc_id - String type; The ID of the document to be searched as the key.

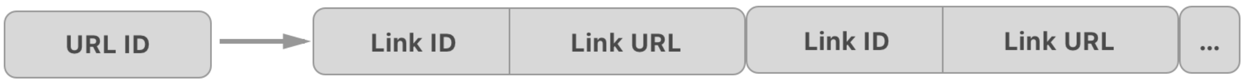
   Return:
   freq - String type; The frequency of appearance of this term in this document.

III. Link Index Table

1. Introduction

Given a URL ID, this table is to store a list of child link IDs and its corresponding child link URL. It provides the relationships between documents by examining the links in between, which could be used in rage ranking.

2. Diagram



3. Implementation

There are two internal functions supported for entry updating and entry retrieving:

1. **appendEntry(url_id, link_id, link)**

    Aims:
    This function is used in Crawler.java in **updateLinkIndex()**, who updates the Link Index table when all the documents is crawled and stored.

    Parameters:
    url_id - String type; The ID of the URL to be inserted as the key.
    link_id - String type; The ID of the link served as the child link of this URL.
    link - String Type; The address string of this child link of this URL.

2. **getEntry(url_id, link_id)**

    Aims:
    It retrieves the child link's content(URL) back as the result, given a specified URL ID and child link ID in the posting list of this parent URL.

    Parameters:
    url_id - String type; The ID of the URL to be searched as the key.
    link_id - String type; The ID of the child link in the posting list of this parent URL.

    Return:
    link - String type; The content(URL) of the child link.

IV. MapTable (Word & URL)

1. Introduction

This MapTable is to build up the mapping relationship between word and its corresponding word ID, as well as URL and its corresponding URL ID. In so doing, it provides bidirectional data/ID retrieving for both word-domain and URL-domain searching.

2. Diagram

Word ID
Word
Word
Word ID
URL ID
URL
URL
URL ID

3. Implementation

There are two internal functions supported for entry updating and entry retrieving:

1. **getEntry(data)**

   Aims:
      It retrieves either the word ID or URL ID back as the result, given a specified word or URL.

   Parameters:
      data - String type; The word or URL to be searched as the key.

   Return:
      word_id or **url_id** - Integer type; The ID refers to the word or URL.

2. **getEntry(id)**

   Aims:
      It retrieves either the word or URL back as the result, given a specified word ID or URL ID.

   Parameters:
      id - Integer type; The word ID or URL ID to be searched as the key.

   Return:
      word or **url** - String type; The word or URL referred by the ID.

3. **appendEntry(word)**

   Aims:
      This function is used in Crawler.java in **updateWordIndex()**, who updates the Word Map Table when all the documents is crawled and stored by inserting a new word-wordID pair and returning the resulted ID.

   Parameters:
      word - String type; The word to be inserted as the key.

   Return:
      word_id - Integer type; The ID refers to this word.

4. **appendEntry(url)**

   Aims:
      This function is used in Crawler.java in **updateLinkIndex()**, who updates the URL Map Table when all the documents is crawled and stored by inserting a new URL-URLID pair and returning the resulted ID.

   Parameters:
      url - String type; The URL to be inserted as the key.

   Return:
      url_id - Integer type; The ID refers to this URL.