**COMP4321 – Search Engines for Web and Enterprise Data**

# Project Phase 1 – Design of Database Schema

Group 44: CHAN, Hau Yan & LI, Tsz Ho

In the project, there are nine databases being used. The first four databases are related to the crawling process, while the latter five are related to the word indexing process. These databases are implemented by JDBM library. Below will define and explain the design of each database. Note that these databases may be changed during the Phase 2 development.

## 1 `urlToPageIdDatabase`

This database maps a URL to a unique Page ID for internal usage later on. The Page ID will keep increaseing when more entries is being added, starting from 0. If a URL is already in the database, it will not be added again.

The key-value mapping is: `url:String ==> pageId:Int`

## 2 `nodePropertyDatabase`

This database stores the properties of webpages, including the URL, title, last modified date and page size. These information are extracted from the `<title>` tag and HTTP header. And the key is the page ID assigned in the previous database.

The key-value mapping is: `pageId:Int ==> properties:HashMap<String,String>`

For the hash map of the `properties`, its keys include `"url"`, `"title"`, `"lastModified"` and `"size"`, with their corresponging values. Note that all the values are converted to String data type before storing.

## 3 `parentToChildDatabase`

This database stores the child URLs of a parent URL based on the webpage content. If there is hyperlink in a webpage, it will be extracted to be the child link. All the URLs are stored as page ID converted by the first databse. Note that multiple same links will only count as one.

The key-value mapping is: `parentId:Int ==> childrenId:HashSet<Integer>`

## 4 `childToParentDatabase`

This database is the inverse of the previous one, storing the parent URLs of a child URL. Therefore, similar extraction method is used and all the URLs are stored as page ID. Multiple same links will also count as one.

The key-value mapping is: `childId:Int ==> parentId:HashSet<Integer>`

## 5 wordIdDatabase

This is a bidirectional database to store a mapping between a word and its ID. Users can retrieve a word by an ID, or vice versa. The word ID will keep increaseing when more words is being added, starting from 0. If a word is already stored, it will not be added again.

The key-value mapping is: `word:String <==> wordId:Int`

## 6 titleInvertedIndexDatabase

This database stores the inverted index of words in title to webpages. After extracting all the words in the `<title>` tag, all the words will be converted to word ID based on the `wordIdDatabase`. And its URL will also be converted to page ID by `urlToPageIdDatabase`. Therefore, the inverted index contains the word frequency of different words in different webpages.

The key-value mapping is: `wordId:Int ==> invertedIndex:HashMap<Integer,Integer>`

The hash map of `invertedIndex` has the page ID as the key, and the word frequency in that particular page title as the value.

## 7 bodyInvertedIndexDatabase

This database is similar to the previous one, where all the words are extracted from the `<body>` tag in the webpages. Same approach is then used for database construction.

The key-value mapping is: `wordId:Int ==> invertedIndex:HashMap<Integer,Integer>`

The hash map of `invertedIndex` has the page ID as the key, and the word frequency in that particular page body as the value.

## 8 titleForwardIndexDatabase

This database stores the forward index of webpages to words in title. Similar method is used as constructing inverted index database.

The key-value mapping is: `pageId:Int ==> forwardIndex:HashMap<Integer,Integer>`

The hash map of `forwardIndex` has the word ID as the key, and the word frequency in that particular page title as the value.

## 9 bodyForwardIndexDatabase

This database stores the forward index of webpages to words in body. Similar method is used as constructing inverted index database.

The key-value mapping is: `pageId:Int ==> forwardIndex:HashMap<Integer,Integer>`

The hash map of `forwardIndex` has the word ID as the key, and the word frequency in that particular page body as the value.