San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

Fall 12-19-2016

Handling Relationships in a Wiki System

Yashi Kamboj
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the Databases and Information Systems Commons, and the Graphics and Human Computer Interfaces Commons

Recommended Citation
Kamboj, Yashi, "Handling Relationships in a Wiki System" (2016). *Master's Projects*. 502.
DOI: https://doi.org/10.31979/etd.z2rt-sfk7
https://scholarworks.sjsu.edu/etd_projects/502

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Handling Relationships in a Wiki System

A Project Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements of the Degree

Master of Science

By

Yashi Kamboj

December 2016

© 2016

Yashi Kamboj

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Master's Project Titled

Handling Relationships in a Wiki System

by

Yashi Kamboj

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2016

Dr. Chris Pollett, Department of Computer Science Date

Dr. Robert Chun, Department of Computer Science Date

Mr. Niravkumar Patel, Blue Jeans Network Date

APPROVED FOR THE UNIVERSITY

Associate Dean Office of Graduate Studies and Research Date

ABSTRACT

HANDLING RELATIONSHIPS IN A WIKI SYSTEM

by Yashi Kamboj

Wiki software enables users to manage content on the web, and create or edit web pages freely. Most wiki systems support the creation of hyperlinks on pages and have a simple text syntax for page formatting. A common, more advanced feature is to allow pages to be grouped together as categories. Currently, wiki systems support categorization of pages in a very traditional way by specifying whether a wiki page belongs to a category or not. Categorization represents unary relationship and is not sufficient to represent n-ary relationships, those involving links between multiple wiki pages.

In this project, we extend Yioop, an open-source content management system, by adding a 'Relationship Handling' system which allows binary representation of relationships. This system uses a novel approach by taking into account the linkages between wiki pages and facilitating search functionality on the basis of these links. Use cases for this feature are genealogy, ontology and dependency identification between multiple entities.

Multiple experiments have been conducted to identify the effectiveness of this feature. In genealogy, the relationship handling system identifies associations between members of a family and provides a clear understanding of family tree. The relationship handling system has also been used to identify dependencies between multiple entities, for

example, dependencies of steps involved in preparing homemade ice cream. This feature helps to understand the structure of the system and navigate between linked entities in a better and efficient way.

ACKNOWLEDGEMENT

This work is dedicated to my parents whose trust in my abilities and unremitting support motivates and energizes me every single day of my life.

I would like to thank my project advisor, Dr. Christopher Pollett, for his enduring patience and constant guidance throughout this project. I would also like to extend my thanks to the committee members, Dr. Robert Chun and Mr. Auston Davis, for their suggestions and time.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Until 2004, wiki systems did not have a dedicated system for organizing articles. The structure was contained in the direct links between the articles, without any tool to organize them further [1]. Starting June 2004, a new feature specifically designed for this purpose was added to all these systems. Category pages, which are basically a collection of links to various articles or other category pages, were introduced. It then became possible to assign articles to category pages, and to link these categories to themselves. The categorization feature and all other operations of wiki systems are built using MediaWiki, a custom-made, free and open source wiki software platform written in PHP, and using the MySQL database system [2].

Categorization is a unary relationship [3]. It involves a single entity i.e. a wiki page and whether it belongs to a category or not. However, there is a no support for n-ary relationships, which refers to links between wiki pages to one or more wiki pages. Thus, implementing a new feature which takes into consideration what relationship type links two pages will be very useful in organizing the content of wiki systems and easy navigation between wiki pages.

This feature is very helpful in certain areas such as genealogy, ontology and dependency identification. In genealogy, identifying which relationship type links which member's wiki page will help in understanding a family tree structure. In ontology, the compartmentalization of the variables and storing the relationships between them can be

easily done with a Relationship Handling (RH) system. The fields of artificial intelligence, the semantic web, software engineering and biomedical informatics, etc. all create ontologies to organize information [4]. In addition to this, a RH system can simply be used to resolve dependencies between multiple entities by exploring the links between one entity to other entities.

The leading question for this project is how wiki structure can be explored efficiently, made easily searchable and navigable on the basis of relationship type between the wiki pages. The category system and possible implementations of categorization has already been investigated [5], but the focus of this project is building a relationship handling system which stores the relationships that link different wiki pages in the database, and using them for graphical exploration of links and facilitating search features.

The most important objects for analyzing the structure of knowledge management system are pages and links. An RH system includes a mechanism for parsing the wiki pages to extract relationship types and pages that any particular wiki page is linked to. This information is saved to the database and further used to explore relationships graphically, be it a ‘binary’ relation or ‘is-a’ relation and provide meaningful search results on the basis of relationship types.

The following chapter, Chapter 2 discusses the previous studies done in the field of structure and organization of wiki structure. Chapter 3 discusses the detailed overview of the proposed RH system. It includes different components which are the essential and necessary source of RH system functionality. Chapter 4 provides an insight into preparation

work done to understand the existing structure of Yioop. Chapter 5 gives an overview about the design and implementation strategy employed for the RH system. It also describes the Wiki Parser, adding relationship to wiki links syntax and regex for filtering those links and storing them to the database for effective information retrieval. Additionally, this chapter also discusses the methods used to look up page relationships on the basis of page name and relationship entered by the user. Chapter 6 discusses the behavior of the RH system as a stand-alone system as well as a system where it is integrated with the Yioop system and its usability to end users. Finally, Chapter 7 discusses areas of improvement and future work.

CHAPTER 2

BACKGROUND

The categorization of wiki systems has both been analyzed and visualized. Holloway et al. [6] compared the top categories and classification structure of Wikipedia 2005 to widely used encyclopedias. A more recent study by Kittur et al. [7] analyzed the growth of categories and developed an algorithm to map wiki articles to the top 11 chosen categories. Although we see that significant research has been done in the field of wiki categorization structure, the exploration of n-ary relationships is a topic that did not attract much attention among the various studies done on the structure of wiki systems.

The current categorization structure of wiki systems has evolved over a long period of time. When categorization was initially added to wiki systems in 2004, there was no mechanism to limit the search results for a particular category [8]. Very large categories caused performance issues, and later changes were made to limit the search results to 200 entries per page. Again, for large categories, users had to navigate through multiple pages in order to see all the results. Thus, page by page mechanism also proved impractical. In mid-2005, the category table of contents was introduced. With the table of contents, it became possible to navigate through large categories with a few clicks. There have been several overlapping views about the purpose of categorization of wiki pages [9]. A few of them are listed below:

1. Categories act as a tool for browsing: These can be used as table of contents, specifying pages linked to a particular category.

2. Category act as a classification system: Categories are a means of classifying wiki pages. Some categories are a subset or a specialized form of another category. Placing a wiki page in just the subset category causes problems in finding all the members of any higher order category.
3. Category act as an index to a subject or other categories: Categories act as an index to a topic that it represents or other categories. It is useful to populate categories at the "level of notability."
4. Categories act as a database search: Categories provide a list of wiki pages that belong to them, hence act as a database search feature.

Implementation of categorization in wiki systems also had various proposals. Some possible implementations are discussed below:

1. Intersection: The proposal was to implement a feature named 'Intersection,' included at every wiki page which displays all other categories this page belongs to. If a new wiki page is constructed and it should also belong to several other categories then links to all those categories are displayed on that page.
2. Manually: Manually prepare categories and add wiki pages to it. This does not seem practical, given the huge size of wiki systems.
3. Pseudo namespace: A "category:" pseudo namespace was proposed along with an automated tool that scans all wiki pages and add links to the database.
4. Field-value pairs: This proposal involved making changes to MediaWiki and including [[category=XYZ]], if XYZ is the category a particular wiki page belongs to.

5. IEG proposal on category system in WikiMedia Foundation (WMF) wikis: This proposal aims at automating the category assignment in wiki systems by developing predictors for categories using metrics and basic statistics.

The present day categorization structure of wiki systems is implemented in a simple way. The MediaWiki software maintains a table of categories, to which wiki pages can be added. Any wiki page can be added to a category by including [[Category: Category Name]] in that page's wikimarkup [10]. There are two tables in the database which store all information about pages assigned to different categories: categorylinks table and category table.

The categorylinks table is used to store information corresponding to all links in the wiki pages. When [[Category: Category Name]] is placed in the wikimarkup of any page, its entry is added to the categorylinks table. The fields in the table are as follows:

1. cl_from: stores the page id of the wiki page where the link was placed.
2. cl_to: stores the name of the desired categories.
3. cl_sortkey: stores the title by which the wiki page should be sorted in the category table.
4. cl_timestamp: Stores the time stamp when the link was created or last updated.
5. cl_sortkey_prefix: This is mostly empty if the page is using the default sort key.
6. cl_collation: This is used to specify collation and if collation changes, run some script to fix rows in database.
7. cl_type: This specifies what type of a wiki page it is: file, subcategory or normal.

```
mysql> describe categorylinks;
+-------------------+-----------------------------+------+-----+-------------------+-----------------------------+
| Field             | Type                        | Null | Key | Default           | Extra                       |
+-------------------+-----------------------------+------+-----+-------------------+-----------------------------+
| cl_from           | int(10) unsigned            | NO   | PRI | 0                 |                             |
| cl_to             | varbinary(255)              | NO   | PRI |                   |                             |
| cl_sortkey        | varbinary(230)              | NO   |     |                   |                             |
| cl_sortkey_prefix | varbinary(255)              | NO   |     |                   |                             |
| cl_timestamp      | timestamp                   | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| cl_collation      | varbinary(32)               | NO   | MUL |                   |                             |
| cl_type           | enum('page','subcat','file') | NO   |     | page              |                             |
+-------------------+-----------------------------+------+-----+-------------------+-----------------------------+
```

Fig. 1. Schema for categorylinks table

The category table tracks all existing categories. Something is considered a category if it exists in the categorylinks table or used to. The fields in this table are as follows:

1. cat_id: This refers to the primary key.
2. cat_title: This refers to the name of the category.
3. cat_pages: This refers to the number of pages in the category.
4. cat_subcats: This refers to the number of sub-categories in this category.
5. cat_files: This refers to the number of files in this category.
6. cat_hidden: This was reserved for future use and removed later.

```
mysql> describe category;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| cat_id      | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| cat_title   | varbinary(255)   | NO   | UNI | NULL    |                |
| cat_pages   | int(11)          | NO   | MUL | 0       |                |
| cat_subcats | int(11)          | NO   |     | 0       |                |
| cat_files   | int(11)          | NO   |     | 0       |                |
+-------------+------------------+------+-----+---------+----------------+
```

Fig. 2. Schema for category table

CHAPTER 3

COMPONENTS OF RELATIONSHIP HANDLING SYSTEM

This chapter discusses the overview of all the processes that are a part of the Relationship Handling (RH) system in Yioop. In order to generate a meaningful response to a user trying to explore wiki relationship, the RH system receives the user inputs and retrieves all relevant wiki pages on the basis of relationship type provided by the user. To be able to effectively perform this, the RH system parses the wiki pages present in its knowledge repository system to extract relationship type between them. However, parsing wiki pages is a complex process and the information contained in wiki pages is much more than just the relationship type and the inter-connections between the pages [11]. The wiki page content includes a lot of other data containing special characters such as html tags, xml tags, hyperlinks, meta-data of webpages and much more information, which is not relevant to the actual information that RH needs. The wiki pages are parsed in a well-structured and articulated manner, and the required information is retrieved using regular expressions developed as part of this project so that the RH system gets to process the valid raw information efficiently.

One of the prominent features of the Yioop system is that it uses MediaWiki's simple text formatting to write wiki pages, also called wiki syntax. In Wikipedia's markup language, free links can be created by putting double square brackets around text designating the title of the page that needs to be linked to. Thus, [[California]] will be rendered as California. Optionally, a vertical bar (|) can be used to customize the link title.

For example, typing [[California|The Golden State]] will produce The Golden State, a link that is displayed as "The Golden State" but, in fact, links to California [10].

3.1 Description of Components

The RH system in Yioop has four main components:

1. Wiki Parser
2. What Links to this Page feature
3. What Relates to this Page feature
4. Advanced Search features in Group Page List

3.1.1 Wiki Parser

Wiki Parser is responsible for parsing MediaWiki documents, both within Yioop, and when Yioop indexes MediaWiki dumps as from Wikipedia to an HTML equivalent.

3.1.2 What Links to this Page feature

The wiki parser of Yioop fetches each individual MediaWiki internal links present in MediaWiki documents to be able to discover connections between different pages. This is done using a regular expression. For example, if page A is related to page B with a link that is displayed as “Page B Link,” then MediaWiki of document A would have [[PageB|Page B Link]]. Thus, the presence of these types of internal links can be extracted using a regular expression. Once we have all the links between the pages, it can be saved to the back end, and later used for information retrieval of linkages between different pages.

3.1.3 What Relates to this Page feature

As discussed in section 3.1.2, the links between different wiki pages are already saved in the database. To delve further into how these pages link to each other, i.e., information about their relationship types, “What Relates To” feature is implemented. It works in conjunction with the wiki parser to extract the relationship type by creating a link extractor regular expression which takes into consideration the relationship type. For example, if page A is related to page B with a link that is displayed as “Page B Link,” and a relationship “Parent,” then MediaWiki of document A would have [[Parent|PageB|Page B Link]]. The fetched values, i.e., relationship type and the linked page are then stored in the database facilitating the search feature based on relationship type.

3.1.4 Advanced Search features in Group Page List

The Group Page list in Yioop displays all the pages in a particular group. In addition to this, a search box is also present which can be used to filter out pages displayed in that group. Another feature has been added here, which enables search on the basis of relationship type and any particular page. The user can choose multiple relationship types and the results would display all transitive relationship results to that particular page.

3.2 Steps to Reproduce the features of Relationship Handling (RH) System in Yioop

Yioop has provided a nice web interface where the user can test these features with a few steps:

1. As a user, login to Yioop as an administrator.
2. After the log in, click on the “Manage Groups” under Social tab.

3. In the Manage Groups section, enter a name for the group and click “Save.”

Fig. 3. Web interface to create a new Group in Yioop

4. Click on wiki link next to the group name. This will take the user to the main page.
5. The user can then edit the page and add links to other wiki pages using MediaWiki language.

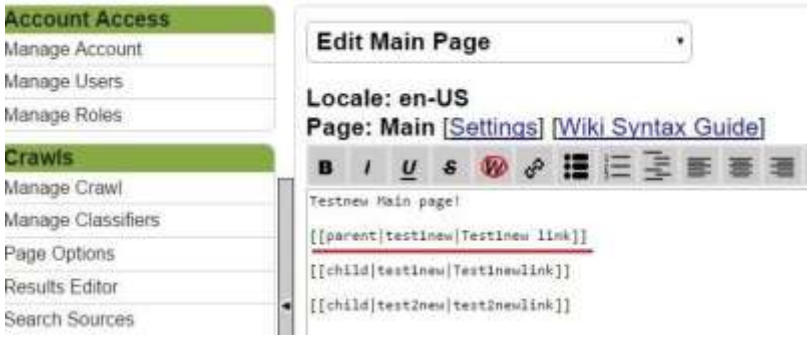

Fig. 4. Web interface to edit a wiki page

6. Similarly, create other wiki pages and connect them with a relationship type and name of the page to which a link has been made.
7. Now click on the drop down menu in the center and select “What Links Here” to see the list of pages that link to any particular page.

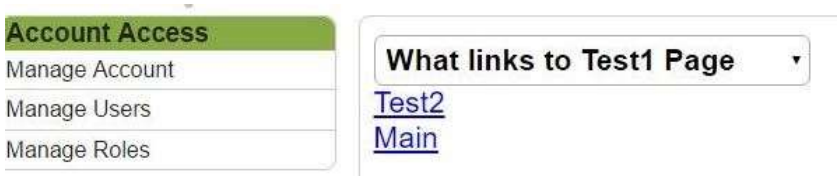

Fig. 5. Web interface to check "What links to the current page"

8. Select "What Relates Here" from the drop down menu to get a list of relationship types that links the current page to other wiki pages.

Fig. 6. Web interface to check "What relates to the current page"

9. The relationship types displayed are hyperlinks and can be explored further to see which wiki pages link "to" the current page with the particular relationship type and which wiki pages link "from" the current page with this relationship type.

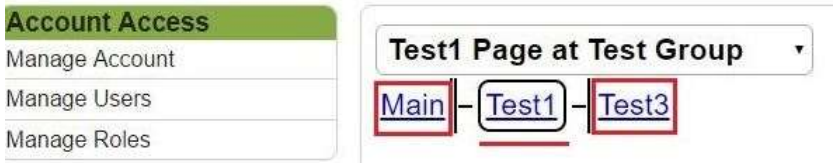

Fig. 7. Web interface to explore relationship types to the current page

10. Click on any other linked page to explore its linked pages.
11. To explore search functionality for transitive relationships, click on the "Group Page List" from the drop down.

12. There is an “Advanced” hyperlink next to the “Go” button. Click on the advanced button to choose relationship type(s) and set value for the page for which we want to explore related wiki pages.

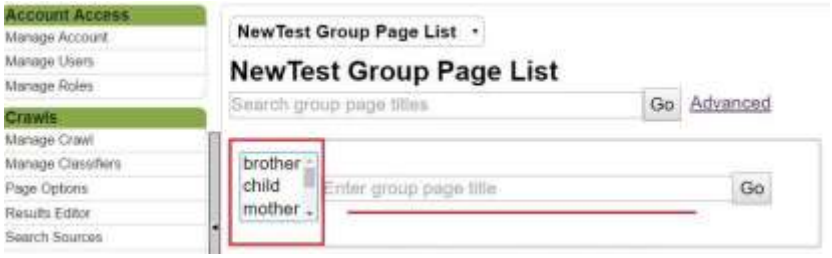

Fig. 8. Web interface to explore search functionality based on relationship type

13. The displayed results can be explored further by clicking on any wiki page in the results.

The RH system comes into the picture after any wiki page with internal links is created and the wiki parser begins to parse the page to fetch all internal links. These links are then parsed further to extract the required data, i.e., relationship type and the linked pages. All the extracted data is stored in the database and then retrieved for exploring wiki page relationships. The technical aspects of the new features added to Yioop will be further explained in Chapter 5.

The image in Figure 9 describes the process flow to handle relationships in Yioop. It also mentions where in the entire process the functionality of each component discussed above comes into the picture.

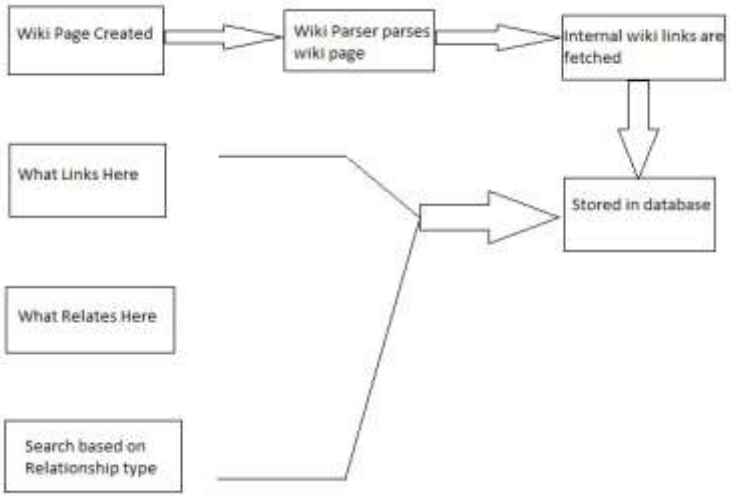

Fig. 9. Overview of information flow

As seen in Figure 9 above, the RH system has three components which pull the required information from the database. The information gets stored in the database while parsing a newly created page or if any wiki page is edited. In addition to this, if a new wiki page is created and has a link to an already existing page, then this case is also handled by updating the links of the pre-existing page.

CHAPTER 4

PRELIMINARY WORK

This chapter discusses the preliminary work done in order to understand the existing wiki system of Yioop. This initial understanding provides a base to implement new features in Yioop for the Relationship Handling (RH) system.

4.1 Resolve an issue present while adding users to the social groups in Yioop

To get a basic understanding of social groups in Yioop, a minor issue while adding users to the groups in Yioop was resolved. Groups are collections of users that have access to a group feed and a set of wiki pages. Groups in Yioop are managed through the 'Manage Group' activity.

4.1.1 Understanding the Issue

The 'Manage Group' activity belongs to the standard user role, allowing any user to create and manage groups. Any user can join a group either by permission or directly. In case membership to a group is granted by request, then the request goes to the administrator of the group who approves/disapproves the request.

There are cases where multiple users request membership to a group. For example, a group created for a class where all students request access to the class group after the very first class. While adding members each time, the user is brought back to the first page of the user list. To save time and effort, the user needs to be on the same page where the user has been added.

The steps to reproduce the issue are as follows:

1. Access Yioop homepage as an administrator.
2. On the left menu bar, go to Social->Manage Groups->click the group in which you got multiple user requests (User requests can also be created by making multiple user accounts and requesting access to the particular group, say "TestGroup.")
3. Click Edit under Actions and then click on the link which displays the number of user requests under Members tab.
4. Click on the link. It displays the number of users, requesting access to the group.
5. Scroll to the next page of the list using ">>". Choose one of the users and activate the user there by clicking Activate.
6. The administrator will be brought back to the beginning of the list.

4.1.2 Solution Approach

The solution will involve passing the information about the page number on which the activation, deactivation, deletion, reinstatement on user is performed by appending it to the URL, while passing the information to the controller about the task to be performed. If the position of the user on which action is performed in the list is saved to a parameter, which is then passed to a method where the actual action is performed then the issue gets resolved.

4.1.3 Implementation

The implementation involved adding the variable, “group_limit” capturing the present position in the user list to the URL and passing it to the controller, where action is performed on the list. When the results were rendered at the user interface, the same position in the list is displayed.

4.2 Understand how existing features present under Manage Groups work

There are multiple features that exist already in Yioop with respect to managing group pages. These involve editing, displaying the page list of all pages in a group, any particular page in a group and its revision history page. To be able to add new features to the social component of Yioop, it is imperative to understand the flow of all these features and how have they been implemented.

4.2.1 Process Flow for each feature

There are multiple features that already exist in Yioop for wiki pages present in social groups. These include displaying a wiki page by parsing its MediaWiki content to HTML, editing a page, history of revisions of a wiki page, discussion, page list and feed of a wiki page. Understanding the implementation of existing features would help a lot to efficiently add new features. So, gaining an understanding of these features was an important part of the preliminary work.

A basic process flow for these features is discussed here. The first step is to add the feature to the drop down menu of all features, selecting which would lead to action related to that feature. The next step is to give a certain term to each value in the drop down menu

which is then passed to the controller along with the page id where we selected the feature. After this, the controller checks the name associated with the drop down and prompts further action based on it. The controller calls the model and passes all required information to get the data. This data is then stored in an array which is then passed to the rendering part of the code. The data is then rendered in the way it is supposed to be presented.

4.2.2 Rendering each feature

Each feature is rendered to present the data that it outputs in a particular way. This part involves front end work and HTML with JavaScript has been used to present the data to the user. The data is stored in an array named $data, extracted and displayed when required.

CHAPTER 5

DESIGN & IMPLEMENTATION OF THE RELATIONSHIP HANDLING SYSTEM

This chapter discusses the design and implementation part of Yioop's Relationship Handling (RH) system. The discussion is divided into two main categories: 1. Design of features added to RH system, and 2. Implementation part, which involves 2.1. Role of the RH system during wiki page creation/edit, and 2.2. Role of RH system during advanced query based on relationship type.

5.1 Design of features added to Relationship Handling system

This section discusses the design decisions taken before implementing features to the RH system. As discussed in section 4.2, some features such as edit a wiki page, save history and display page lists related to a social group already exist in Yioop. To add new features, it is imperative to follow the design approach used in the existing features to maintain uniformity of code.

While adding any new feature, the triggering point is that the user selected that particular feature from the list of drop down menu options available. On selecting any new feature, a new page shall open up for the display of results. All the computations are done in between the two steps mentioned above. All the data that the user provided shall append to the URL, and then passed on to the controller which is responsible for interaction with the database. Design decisions involved how to fetch the values that the user provided and pass them on to the controller for performing computations, and utilizing all resources and

information available. Additionally, some tables have been created to store relationship links between pages.

Results are displayed in an easier to comprehend manner. Rendering data to the front for each feature is a creative task. In addition to this, displaying results as hyperlinks for further exploration of any feature help users explore transitive relationships.

5.2 Implementation of features added to Relationship Handling system

This section discusses the implementation part of features which form part of the Relationship Handling system. It is further divided into two parts: Role of RH system during wiki page creation or editing, and role of RH system during advanced search based on relationship type.

5.2.1 Role of the Relationship Handling (RH) System during wiki page creation/edit

As mentioned in Chapter 3, the primary objective of the wiki parser with respect to the RH system is to parse the wiki pages and retrieve the internal links from the MediaWiki content of the page. Whenever a new wiki page is created, it is parsed and the useful data is retrieved and stored. This has already been discussed in section 3.1.1. Now, the implementation of each feature forming a part of RH system will be discussed.

5.2.1.1 Implementation of "What Links Here" feature

Before understanding its implementation in Yioop, we shall look at other wiki systems and understand how it is being implemented there. "What Links Here" feature is used to list all other pages which link to a given page. In existing wiki systems, this is implemented by making a wikilink using this syntax: [[Special:WhatLinksHere/Page

Name]], where page name refers to the name of the current page. This is implemented using two tables: *pagelinks* and *templatelinks* table. Pagelinks table tracks all internal links in the wiki. Each entry in this table contains page id of the source page, its namespace (number), name (in text), and namespace (number) that is being linked to from the current page. The templatelinks table does the same function with template wiki pages.

To implement this feature in Yioop, the first step is to obtain all wiki pages that are connected to other wiki pages. Secondly, obtain page ids of all the linked pages. These values are then stored in the database. The starting point is to check the presence of links where the WikiMedia content is transferred to HTML. Store all the fetched links to an array and then get the page ids for all linked wiki pages. The regular expressions used to obtain links from WikiMedia content are as shown in Figure 10. These links are converted to HTML by using the anchor (<a>) tag. Further discussion on first substitution, present in Figure 10, will be done in "What Relates Here" in section 5.2.1.2 Here, $base_address refers to the base URL used for link substitutions.

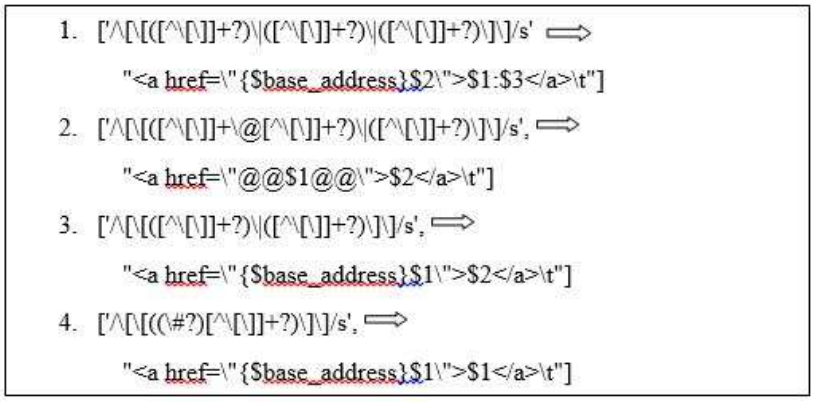

Fig. 10. Regular expressions to fetch links in a MediaWiki document

function LINK-EXTRACTION(Wiki page) returns all links present, or null

result ← MATCH-REGEX(MediaWiki content)
EXTRACT-LINKS(MediaWiki content)
OBTAIN-PAGEIDS(Extracted_links)

if result is not null then store result in database and return
else return null

Fig. 11. Function to extract links in a MediaWiki document

The table that stores this information is *GROUP_PAGE_LINK: CREATE TABLE GROUP_PAGE_LINK (LINK_TYPE_ID INTEGER, FROM_ID INTEGER, TO_ID INTEGER). Additionally, there is a table PAGE_RELATIONSHIPS: CREATE TABLE PAGE_RELATIONSHIPS (ID INTEGER PRIMARY KEY AUTOINCREMENT, NAME VARCHAR(32) UNIQUE).*

5.2.1.2 Implementation of "What Relates Here" feature

This feature lists all relationship types with which a page is connected. Consider page A which is parent to page B, and child to Page C. Then Page A shares two relationships: parent and child. This feature displays relationship types as a hyperlink. These hyperlinks provide further information on which pages connect "to" the current page and which pages connect "from" the current page with that particular relationship type. This has been explained already in section 3.1.3. To implement this, the first step is to obtain the page id of the current page and then obtain pages connected to the current page in two different arrays: connected to the current page and connected from the current page. In addition to this, get a list of relationship types with which it is connected by joining the *GROUP_PAGE_LIST* table with *PAGE_RELATIONSHIP* table.

```
function RELATIONSHIP-EXTRACTION(Wiki page name) returns all
relationship types connecting to and from a given page, or null

    page_id ← OBTAIN-PAGEIDS(Page Name)
    To-result←EXTRACT-LINKED-TO-PAGES & Relationship(Page Id)
    From-result←EXTRACT-LINKED-FROM-PAGES & Relationship(Page
    Id)
    relationship_names←Unique(OBTAIN-RELATIONSHIP-TYPE-
    NAMES(To-result, from-result))

    if relationship_names is not null then
       display each relationship-type
       return relationship_names
    else return null
```

Fig. 12. Function to extract relationship type connecting a wiki page

To further explore the relationship type displayed as a hyperlink, click on any hyperlink which opens a new page with a tabular structure providing a list of wiki pages that the page is connected *to* towards the left and a list of wiki pages connected *from* towards the right with the particular relationship. In the middle, the link to current page is displayed. Each wiki page displayed on the new page, in turn, can be explored further in the same way just by a single click.

5.2.1.3 Implementation of "Advanced Search" feature:

This feature expands the search functionality to implement a search based on relationship type, as discussed in section 3.1.4. The advanced hyperlink opens up a new form where multiple relationship types can be chosen by the user and a page name can be provided from where we want to explore transitive relationships of the types chosen.

The first step is to get a list of all relationship types existing in the system and display them with a text box for the user to enter a page name. The page id of the entered

page name is obtained and with all the relationship types chosen by the user, a list of related pages is obtained and displayed. This list of page names can be further explored to find their transitive relations with the already selected relationship types. The function implementing it can be described as in Figure 13.

function ADVANCED-SEARCH(Wiki page) returns search results based on relation type, multiple selection allowed, or null

relationship-types ← OBTAIN-ALL-RELATIONSHIPS()
page_id ← OBTAIN-PAGEID(Wiki Page Name)
linked-pages ← OBTAIN-LINKED-PAGES(page_id, relationship-types)

if linked-pages is not null then
add hyperlink to each linked page, return linked_pages
else return null

Fig. 13. Function to implement advanced search features

All wiki pages returned as a result can also be explored further for transitive relationship with the relationship types selected.

5.2.2 Role of Relationship Handling (RH) System during Advanced Query based on Relationship Type

During advanced search time, the user inputs a page name and derives a transitive relationship result for all relationships chosen. The RH system adds a search feature to Yioop based on relationship(s). This search returns results for all wiki pages linked to the chosen page based on relationship type(s) selected. The transitive results are returned by the RH system and these results can also be further explored for relationship links.

In order to process this search, the RH system appends all the required values to the URL and then provides it to the controller. The values are fetched from the request variables present in the URL by the controller and the appropriate actions taken by interacting with the database. Similar approaches are discussed in section 4.2.

CHAPTER 6

EXPERIMENTS

This chapter discusses the experiments carried out on the developed Relationship Handling (RH) system. These experiments are divided into three parts:

1. Experiments conducted to test the working of all new features added to RH system
2. A/B Testing
3. Experiments conducted on use-case scenarios of RH system

6.1 Experiments conducted to test the working of RH system

The experiments conducted on the RH system give an idea about the working of the system under ideal conditions. Here, an assumption is made that the content added to wiki pages adheres to MediaWiki syntax.

The RH system works on the convoluted structure of wiki pages linked to each other with certain relationship types. The experiments will be conducted by assuming an example of a social group in Yioop with certain wiki pages linked to each other. Figure 14 explains the structure of wiki pages that we will be experimenting on. The structure mentioned in Figure 14 can be tabulated as done in Table I.

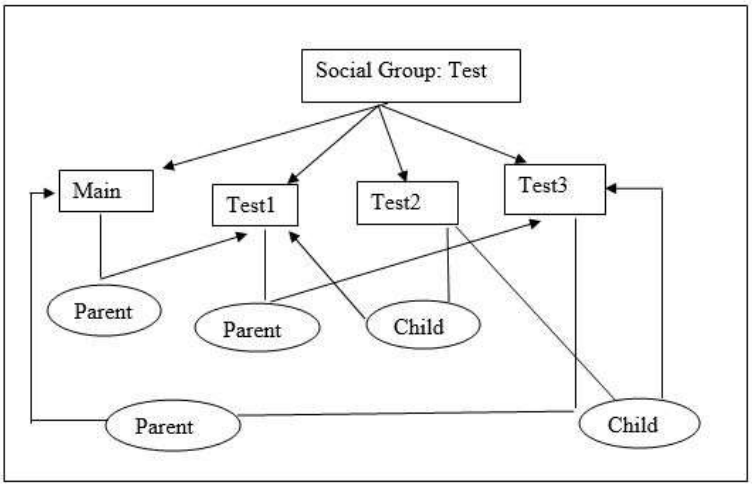

Fig. 14. Relationships between wiki pages depicted

Table I
LINKS BETWEEN WIKI PAGES USED FOR CONDUCTING AN EXPERIMENT

Relationship Type	*From Page*	*To Page*
Parent	Main	Test1
Parent	Test1	Test3
Child	Test2	Test1
Child	Test2	Test3
Parent	Test3	Main

One of the few experiments carried out on the system is for checking the links between all the pages. Before proceeding further, content has been added to all the wiki pages. These pages contain links to other wiki pages using the MediaWiki link syntax. A few changes have been made to that syntax to be able to incorporate the relationship type as discussed in section 3.1.3.

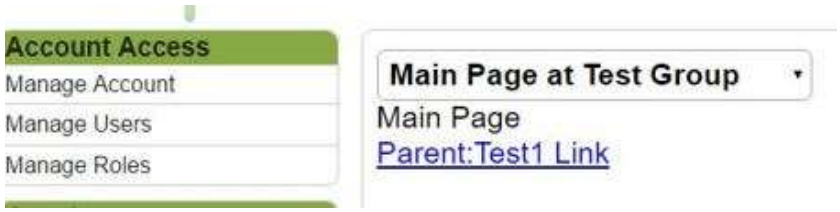

Fig. 15. Content on Main page of Test group

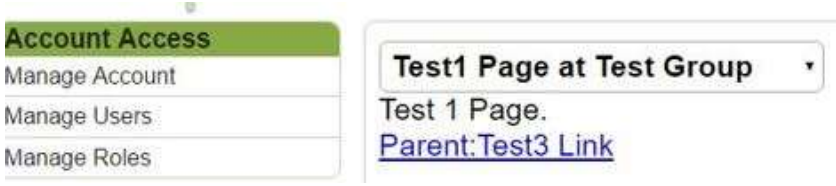

Fig. 16. Content on Test1 page of Test group

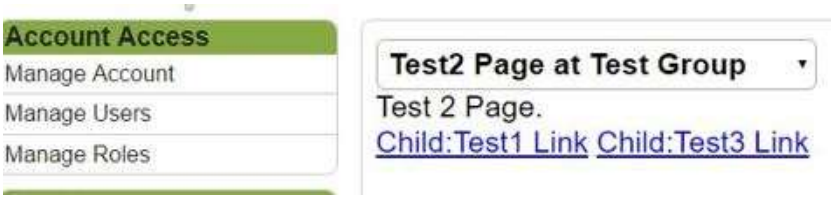

Fig. 17. Content on Test2 page of Test group

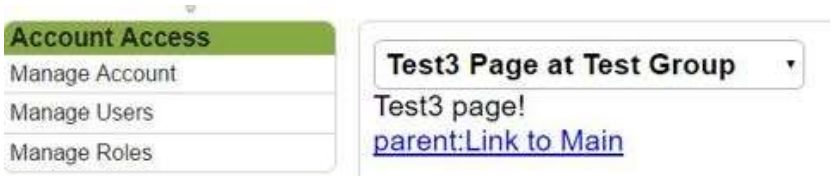

Fig. 18. Content on Test3 page of Test group

Example 1:

To understand links between these wiki pages, we can select “What Links Here” from the drop down menu of all actions that can be performed on any wiki page of a social group. The structure explained in Figure 14 shows that the Main page is linked to Test3

page (from its MediaWiki content). To verify this, “What Links Here” feature is used, as shown in Figure 19.

Fig. 19. “What Links to” Main page

As we can see in Figure 19, the result is a hyperlink to the page displaying content of Test3 page. Similar experiments for other pages appear in Figure 20, 21 and 22.

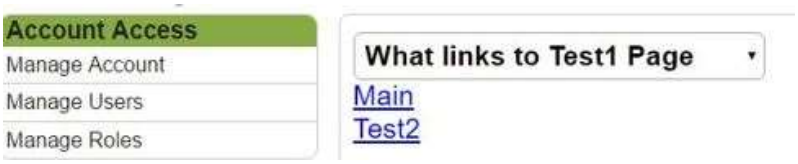

Fig. 20. “What Links to” Test1 page

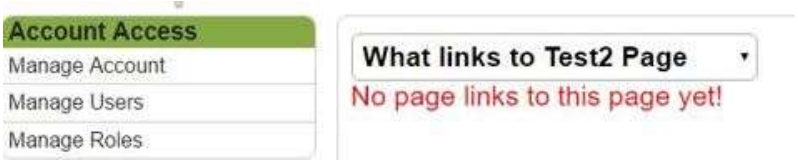

Fig. 21. “What Links to” Test2 page

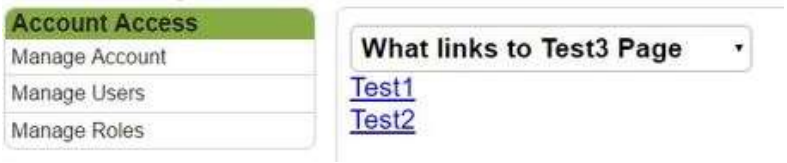

Fig. 22. “What Links to” Test3 page

Example 2:

To understand the relationship types with which any wiki page is connected, “What Relates Here” feature has been implemented. It takes into consideration both “To” and “From” relationships, as discussed in section 3.1.3. For more clarity, refer Figure 7.

Fig. 23. “What Relates to” Test3 page

Similar action can be performed for other wiki pages. The relationship types in Figure 23 can further be explored to specifically check which pages links “To” and which pages link “From” the particular wiki page, (Test 3 in this case). Figure 24 analyses relationship types in more detail.

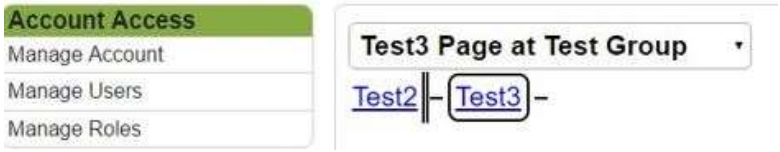

Fig. 24. Pages linked To & From Test3 page with Child relationship

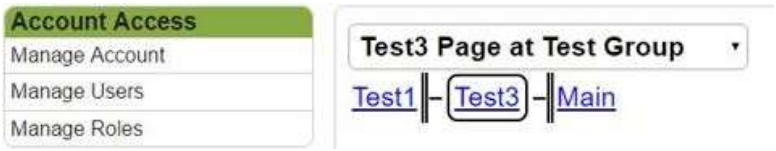

Fig. 25. Pages linked To & From Test3 page with Parent relationship

Similar action can be performed for other wiki pages. The results to the left of the wiki pages are the ones which connect to that wiki page and the ones to the right connect

from that wiki page, as discussed in section 3.2. If multiple pages link to or from the wiki page, then a list of those pages appear as shown in Figure 26.

Fig. 26. Pages linked To & From Test2 page with Child relationship

Example 3:

The advanced search feature in Yioop performs a search on the basis of relationship types, as discussed in section 3.1.4. The user interface for advanced search would look like as shown in Figure 27.

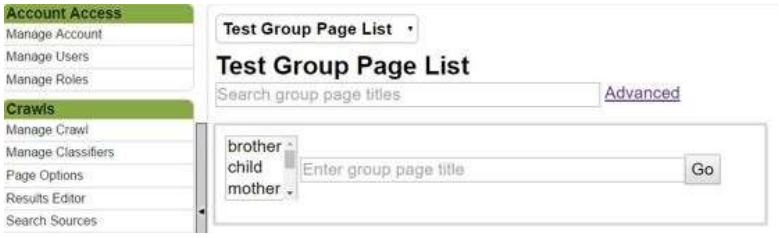

Fig. 27. Advanced search performed on the basis of relationship type

6.2 A/B Testing

This testing, also known as split testing, involves comparing two different versions of a web page to see which one performs better. The two variants are shown to similar visitors and their responses are recorded. The one that provides a better conversion rate, wins.

This testing was conducted by comparing two different versions of the advanced query webpage. The difference in the two variants was about placing a single text box for users to enter the page name. In the first version (variant A), only one text box is present, which records the page name and outputs all pages that are linked to that page or linked from that page, as shown in Figure 27. On the other hand, the second variant (variant B), shown in Figure 28, provides two text boxes to the user, one for entering a page *from* which links will be fetched and second for entering a page *to* which all links will be fetched.

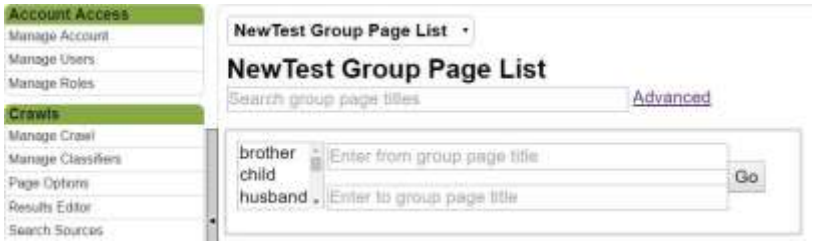

Fig. 28. Variant B for the advanced search feature

Both the variants were shown to three different users and their opinions were recorded as below:

User 1: This user initially found it difficult to understand what is meant by “from” group page and “to” group page name. Once it was explained, the user found variant B to be better, providing more flexibility to the user.

User 2: This user believed that variant A is much simpler and easier to use. The user need not go into the details of how the links are stored in the form of to-links and from-links. From a user’s perspective, variant A is better by simply letting the user type a

page name and get all results for pages linked to and from that particular page with the given relationship type.

User 3: The third user thinks that variant B is unnecessary complicating things from the user's perspective and if a user wants to get the links for multiple pages then this feature can be re-used. Also, if a user wants to get the "from" and "to" links for the same page then it is not a good idea to provide separate text boxes to enter the same page name.

The opinions of these users indicate that the simpler the system is to the user, the better it is. Although variant B provides more flexibility to the user, it brings along complications and confusions from the user's perspective. This system shall keep "from" and "to" link storage at the database transparent from the user and proceed with variant A.

6.3 Experiments conducted on use-case scenarios of RH system

The system has been used in real-life situations as genealogy and resolving dependencies between components using this feature. The feedback has been taken from the users and changes have been incorporated. The use-case scenarios for the Relationship Handling (RH) system in Yioop are as follows:

1. Genealogy experiments
2. Topological sort – Dependency Resolution experiments

Example 1:

To test the Question Answering system application in real world, a genealogy example has been used. The family tree used for experimentation has been shown in Figure 29 and its tabulated version in Table II.

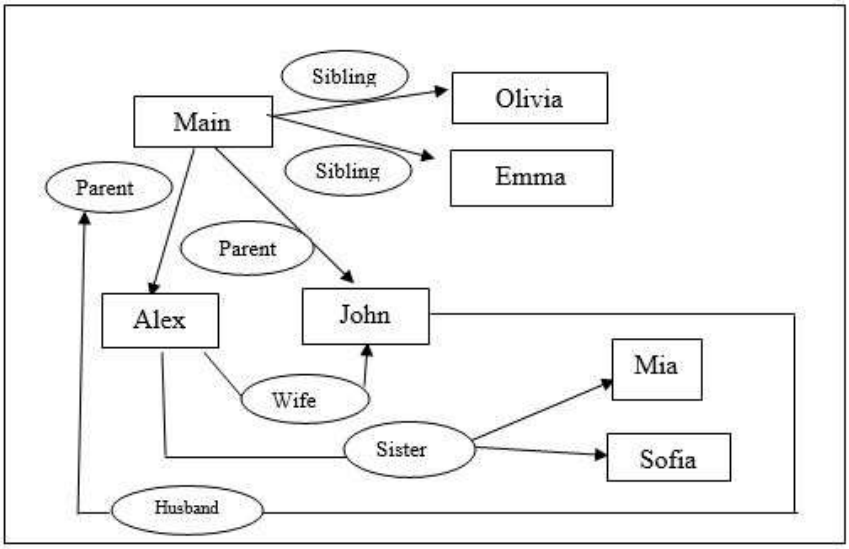

Fig. 29. Family tree used as a use case for the Relationship Handling system

Table II
DEPICTS RELATIONSHIP BETWEEN MEMBERS OF A FAMILY

Relationship Type	*From person*	*To person*
Parent	Main	Alex
Parent	Main	John
Sibling	Main	Olivia
Sibling	Main	Emma
Husband	John	Alex
Wife	Alex	John
Sister	Alex	Mia
Sister	Alex	Sofia

To see who is related to Alex, we go to Alex's wiki page and check "What Links Here." The results are shown in Figure 30.

Fig. 30. Alex's wiki page links

In addition to this, all relationships Alex is related to other people in the family is shown in Figure 31.

Fig. 31. Alex's wiki page relationships

These hyperlinks can further be explored to see who connects to Alex with a particular relationship and with whom Alex connects. For example, Joey is husband to Alex and Alex is wife to Joey. Figure 32 and 33 depict this as shown below:

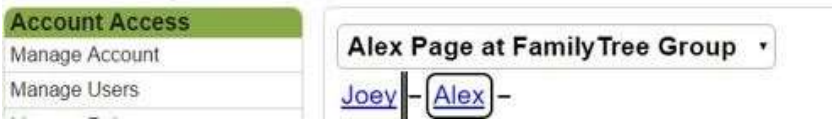

Fig. 32. Joey's relationship to Alex

Fig. 33. Alex's relationship to Joey

Members in the family tree can also be searched on the basis of relationship type. For example, search all siblings of Alex. Then all first siblings to Alex will appear. Also, Alex's first sibling's siblings will also be Alex's siblings, so they will also form part of the result.

Example 2:

To further test the usability of RH system, we consider another use-case scenario in which our system would act as a dependency resolver. If a project has many components and each component depends on other components for its execution (assuming resolvable dependencies), then the component with no dependency would be executed first, followed by components that depend on components already executed and so on. This is an application of topological sort, i.e., sorting the dependencies on the basis of their dependencies over other components of the project. Figure 34 explains the dependency structure of the project under discussion:

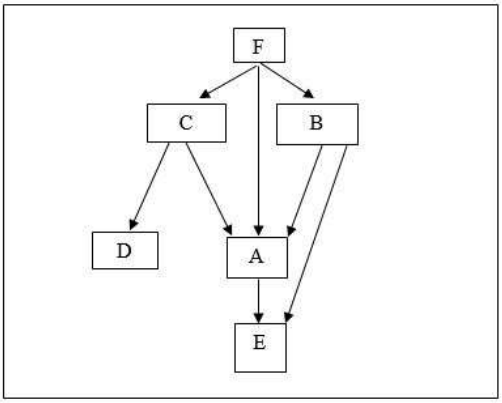

Fig. 34. Dependencies among the components of a project

To check which component should be executed first, one can explore the relationship tab and see what links “from” and “to” all project components. If a component depends on no other component, i.e., there is no project with which a particular project is linked to, then it can be executed first. This step is followed by removing the dependencies of this component with all other project components by removing it from the MediaWiki of those pages. It can be also be scheduled to initialize deletions related to a particular project component from the database directly as soon as that project is executed. Proceeding the same way, dependencies of all projects can be resolved.

CHAPTER 7

CONCLUSION

In this project, a new module called Relationship Handling system is developed for Yioop. This module is responsible for storing the information in a manner such that it can be later used for understanding the links between wiki pages and searching based on relationship types. This system employs an approach of storing the relationship types and links between wiki pages in the database to be used for exploring relationships and facilitating search features.

The RH system processes the MediaWiki content present on wiki pages. It handles the relationship type, specified to link one wiki page to other wiki pages and stores the result at the Yioop's database. This system also takes care of any search implemented to check transitive relationships. This system will help Yioop users to understand the overall structure of database by getting an understanding of links between wiki pages.

Currently, the components of the RH system are unable to understand if a particular relationship exists from A to B, then some kind of relationship also exists from B to A, unless specified. Consider a genealogy example where A is B's brother and B is female, this implies B is A's sister. In our system, after specifying A is linked to B as a brother, it also needs to specify separately that B is linked to A as a sister. Future work can be done to improve this system so that such cases can be handled, thereby, saving time and effort, and increasing the overall efficiency of the system.

CHAPTER 8

REFERENCES

[1] MediaWiki links. (2015). Retrieved from https://www.mediawiki.org/wiki/Help:Links#Internal_links

[2] M. Bergman, "Wikimedia architecture" (PDF). Wikimedia Foundation. [Accessed 27-Sep-2016].

[3] arXiv.org, "Evolution of Wikipedia's category structure," 2012. [Online]. Available: https://arxiv.org/abs/1203.0788. [Accessed: 20- Nov- 2016].

[4] Ontology. (2016). Retrieved from https://en.wikipedia.org/wiki/Ontology_(information_science).

[5] L. Muchnik, R. Itzhack, S. Solomon, and Y. Louzoun, "Self-emergence of knowledge trees: Extraction of the Wikipedia hierarchies," Physical Review E 76, 2007.

[6] T. Holloway, M. Bozicevic, and K. Borner, "Analyzing and visualizing the semantic coverage of Wikipedia and its authors," *Complexity* 12, 2007, pp. 30–40.

[7] A. Kittur, E. H. Chi, and B. Suh, "What's in Wikipedia? mapping topics and conflict using socially annotated category structure", in *Proc. 27th Annual CHI Conference on Human Factors in Computing Systems (CHI'2009)*, New York, USA, 2009, pp. 1509–1512.

[8] “Announcement of Wiktionary's creation,” meta.wikimedia.org. [Accessed 10-10-2016].

[9] arXiv.org, “Connecting every bit of knowledge: The structure of Wikipedia's first link network,” 2012. [Online]. Available: https://arxiv.org/abs/1605.00309. [Accessed: 05- Nov- 2016].

[10] Wiki markup. (2015). Retrieved from https://en.wikipedia.org/wiki/Help:Wiki_markup.

[11] Seekquarry.com, “Resources,” 2015. [Online]. Available: https://www.seekquarry.com/p/Resources. [Accessed: 09- Sep- 2015].