AUTHORITY CONTROL OF ROMANISED ARABIC NAMES IN ONLINE PUBLIC ACCESS CATALOGUE

Thesis submitted to University of Calicut

for the Degree of

DOCTOR OF PHILOSOPHY

IN LIBRARY AND INFORMATION SCIENCE

By

VEERANKUTTY CHELATAYAKKOT

Under the Supervision of

Dr. V. JALAJA
Reader and Head

DEPARTMENT OF LIBRARY AND INFORMATION SCIENCE

UNIVERSITY OF CALICUT

2005

DEPARTMENT OF LIBRARY AND INFORMATION SCIENCE

UNIVERSITY OF CALICUT

Dr. V. JALAJA
Reader and Head

Calicut University P.O
Malappuram Dist.
PIN 673 635

CERTIFICATE

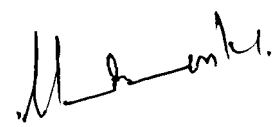
I certify that the thesis entitled "**Authority Control of Romanised Arabic Names in Online Public Access Catalogue**" is record of bona fide study and research carried out by Sri Veerankutty Chelatayakkot under my supervision and guidance. This thesis is submitted to University of Calicut for the award of the Degree of Doctor of Philosophy in Library and Information Science.

Dr. V Jalaja
Supervising Teacher

Thenjippalam
4th July, 2005

DECLERATION

I, Veerankutty Chelatayakkot, do hereby declare that this thesis "**Authority Control of Romanised Arabic Names in Online Public Access Catalogue**" has not been submitted by me for the award of any Degree, Diploma, Title, or Recognition before.



VEERANKUTTY CHELATAYAKKOT

4th July, 2005
C. U Campus,

ACKNOWLEGEMENT

Alhamdu Lillah

It is my privilege and great pleasure to acknowledge all those who have provided help, inspiration and encouragement in the completion of this academic endeavor directly or indirectly. With great reverence, I express my heartful thanks to my Supervising Teacher Dr. V. Jalaja, Reader and Head, Dept. of Library and Information Science for her valuable guidance, inspiration for the completion of this research.

I express my sincere gratitude to Prof. M Parameswaran, former Head of the Dept. of Library and Information Science, University of Calicut for his valuable assistance and encouragement at every stage of this work.

I take this opportunity to convey my sincere thank to Sri. Hussain. Ahmad P, Software Engineer, Tata Alexi, Technopark, Thiruvananthapuram for his assistance in fulfilling the work.

I extent my sincere thanks to Dr. K.V.Promod, Lecturer, Dept. of Computer Application, Cochin University of Science and Technology, for his immense support. I also thank to Dr. S Babu Sunder, Head, and my colleagues for their support from the department.

I acknowledge my thanks to office staff of Department of Library and information Science, University of Calicut.

I owe my gratitude to my parents and family members for their whole hearted support.

VEERANKUTTY CHELATAYAKKOT

CONTENTS

LIST OF ABBREVIATIONS

AACR	: Anglo American Cataloguing Rules
ALA	: American Library Association
ANN	: Artificial Neural Network
ASCII	: American Standard Code for Information Interchange
ASR	: Automated Speech Recognizer
CLIR	: Cross Lingual Information Retrieval
DBLP	: DataBase Systems and Logic Programming
DTD	: Document Type Definition
ICCP	: International Conference on Cataloguing Principles
ISAN	: International Standard Author Number
ISADN	: International Standard Authority Data Number
ISBD	: International Standard bibliographic Description
LCNAF	: Library of Congress Name Authority File
LCRT	: Library of Congress Romanization Table
LCSH	: Library of Congress Subject Heading
MARC	: Machine Readable Catalogue
NACO	: Name Authority Cooperative
NAF	: Name Authority File
OPAC	: Online Public Access Catalogue
OOV	: Out Of Vocabulary
RLG	: Research Library Group
XML	: eXtensible Markup Language

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The story of catalogue and cataloguing is one of the aspects of the broad panorama of library development. Earlier, libraries concentrated their efforts on the collection and preservation of manuscripts. Later, after the invention of printing, libraries started collecting and organizing materials, which would be kept for reading, study and consultation. For easy location and consultation, librarians and libraries used some system of bibliographical organization or control over the reading materials. This bibliographical organization, in the context of library, is called library catalogue.

S.R. Ranganathan[1] defines library catalogue as "a list of documents in a library or in a collection forming a portion of it." It implies three concepts of library catalogue, i.e., (i) list, (ii) document, and (iii) holding. In 1876, Charles Ammi Cutter[2] clearly defined the objectives of library catalogue. The objectives are as follows.

"1) to enable a person to find a book of which
   a) the author or,
   b) the title or,
   c) the subject is known.

2) to show what the library has
   d) by the given author,
   e) on a given subject,
   f) in a given kind of literature.

3) to assist in the choice of a book
   g) as to its edition,
   h) as to its character"

The objectives put forward by Charles Ammi Cutter is still being quoted even though the physical form of library catalogue has changed from card format to electronic format, i.e., Online Public Access Catalogue (OPAC). The advent of computer and

communication technology has drastically changed the arena of cataloguing. A computer itself can also be used as a catalogue i.e., the information can be stored within the computer and kinds of entries required can be re-produced as when required.

Cutter's objectives of catalogue have remained constant, even while the means of achieving them have changed with the evolution from book, through card, to online catalogue. Cutter's objectives were largely adopted as the functional framework for formulating the set of cataloguing principles known as "the Paris Principle" that emerged from International Conference on Cataloguing Principles (ICCP) held at Paris in 1961. The ICCP described the functions of a library catalogue in the following words:

> "The catalogue should be an efficient instrument for ascertaining
> 1. whether the library contains a particular book specified by
> a) its author and title, *or*
> b) if the author is not named in the book, its title alone, *or*
> c) if author and title are inappropriate or insufficient for identification, a suitable substitute for the title; and
> 2. a) which works by a particular author and
> b) which editions of a particular work are in the library."[3]

The second objective put forward by Cutter and the second function of catalogue described by ICCP implies the importance of collocation function. Collocation function means bringing together all the entries having same heading in a catalogue. If author uses variant names, cross-references are needed to connect his/her variant names.

The selection of one form of heading as entry heading and creating cross-references from other headings was the usual practice of the cataloguer. In the days of book and card catalogue, the cataloguer maintained a record (authority record) of their decision for the standard form of a heading and the variant forms for which cross reference entries were made. These records were mainly needed for larger catalogue and cataloguing units to maintain consistency among multiple cataloguers. The authority control helped the cataloguer to prepare entries of all the works of an author under one standard form of

name and provide references to this form of name from variant forms. This procedure saves the time of the users since he can get all the works of an author under one standard form of heading. The reference directs him to this heading from variant forms of names.

The authority records documented the catalogers' authority work. Authority work consists of "the creation of authority records, the formation of such records into an authority file, the linking of that file to the bibliographic file to form a system, the maintenance of the authority file and system, and the evaluation of the file and system."[4]

Authority control of a library catalogue is maintained through an authority file that contains the terms used as access points in the catalogue. The access points that determine the structure of the catalogue may be real entry headings on bibliographic records or cross-references. In library catalogue, the entry headings under control generally consists of personal and corporate names, uniform titles, series and subjects.

Authority control can provide:

a) *consistency* which is particularly important if a library obtains its cataloguing from many different sources;

b) *collocation* of records with the same access points which in turn helps in the formulation of effective search strategies resulting in good information retrieval;

c) *cross reference* structure showing relationships among controlled headings and

d) local policy on standardization of headings which contributes to the overall utility of the catalogue.

On the cataloguer's perspective, the functions of authority control are:

a) *Authority function*:- like systematic organization, libraries want to establish consistency in the nomenclature used in their catalogue. Achieving this consistency is extremely time consuming and requires highly trained staff;

b) *Finding function*:- taxonomist refer to different spellings, synonyms, homonyms hierarchies and so on. The idea is to record these somehow, and link them to authorized nomenclature. Cataloguers do the same thing when they provide references from variant or related forms of a name;

c) *Information function*:- authority records usually contain documentation about the sources to establish the name or subject heading, and

d) *Maintenance function*:- authority data is also used to support the detection and correction of errors in library catalogue.

Authority Control in the Card Environment

Learning about authority control in the card environment gives a better understanding of authority control principles, which can be applied to the automated environment.

Examples of a name authority in card environment; (Choices and forms of entry for names are based on rules set forth in AACR-2)

Key to an Authority card

x = Make See reference from

xx = Make See also reference from.

Authority record Card

Le Carre, John

x Cornwell, David John Moore, 1931-

x LeCarre, John, 1931-

x Carre, John Le, 1931-

823

L461

Figure 1.1: Format of Authority Control Record

Card Catalogue (what the public see)

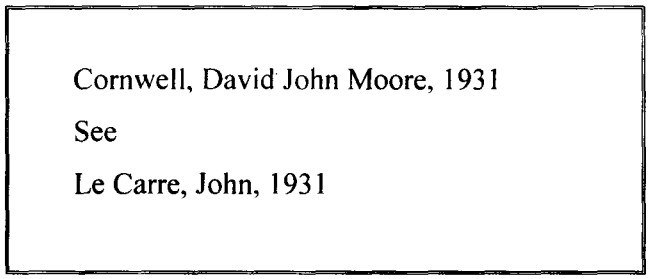


Figure 1.2(a)

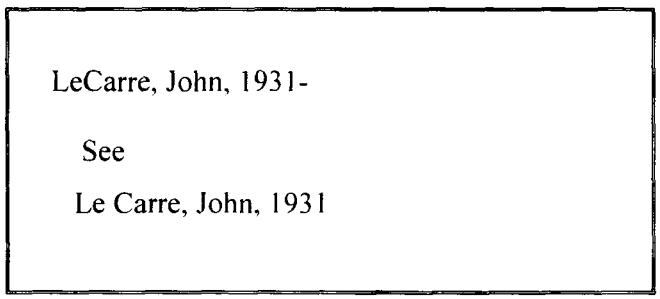


Figure 1.2(b)

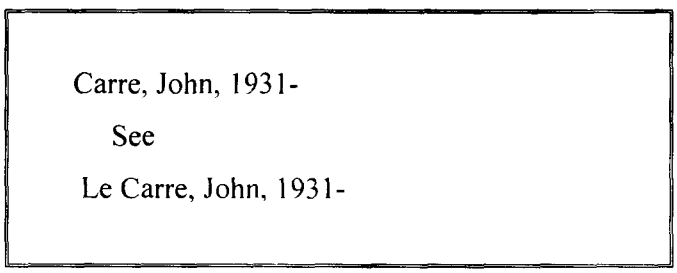


Figure 1.2(c)

Figure 1.2 (a),(b),(c): Formats of See References

Authority control in the Automated Environment

Authority control is used to establish standardized key access points and references to ensure effective access to library catalogue. In the automated environment, MARC records are created for authorities and serve the same function as in the card environment. Authority record in MARC format has one fixed field that contains coded information and many variable fields that usually contain textual information.

For example,

0xx : Call numbers , control numbers

1xxs: Established headings

260: Complex See Reference (Subject)

360 : Complex See also Reference (Subject)

4xxs: See Reference

5xxs: See also Reference

6xx: Notes, series treatment information.

7xx: Heading linking entries.

```
001      n  50014331
003      DLC
005      19950123115723.1
008      800410n| acannaab|a aaa |||| cz n
010      n  50014331 |zn  89119817
040      DLC|cDLC|dDLC
053      PJ7864.A35
100   10 Husayn, Taha,|d1889-1973
400   00 Taha Hussein,|d1889-1973
400   10 Hussein, Taha,|d1889-1973
400   00 |wnna|aTaha Husayn,|d1889-1973
400   10 Husain, Taha,|d1889-1973
400   10 Huseyn, Taha,|d1889-1973
400   10 حسين, طه,|d1889-1973
670      His Kudama ibn Jafar, Abu al-Faraj, al-Katib
           al-Baghdadi,|bNakd an-Nathr ... 1933.
```

Fig. 1.3 Format of MARC Authority Record

Online Public Access Catalogue

ALA Glossary defines OPAC as "A computer based and supported library catalogue (bibliographic databases) designed to be accessed via terminals so that library

users may directly and effectively search and retrieve bibliographic records without the assistance of human intermediary such as a specially trained member of library staff."[5]

According to Gorman, OPAC is " a bibliographic control system that allows access by means of access points conventional and unconventional, single and combinational form. The data retrieved is displayed on a terminal screen or printed out on demand. Terminals are housed in the library or elsewhere."[6]

So, OPAC is an access tool and resource guide to the collection of a library or libraries, which provides bibliographic data in machine-readable form and can be searched interactively on a computer terminal by users. Any OPAC should have some of the qualities given below:

a) OPAC is a bibliographical control system;

b) It allows search by a number of access point to the bibliographic data stored in machine-readable form;

c) It provides instrumental help;

d) It displays search result in relatively understandable form, and

e) It is an interactive information retrieval system.

OPACs began to appear in libraries in early 1980's. Many OPACs represent a radical departure both from card catalogue and from offline computerized catalogue in their implication for the users. Current advances in computer technology and distributed networking have contributed to a surge of interest in the development of a new generation of OPACs that are ultimately intuitive and require a minimum of instruction. Remote access through Internet, the client/server model, and standards such as NISO, Z 39.50 has opened new doors for vendors and researchers to propose innovative OPACs.

The World Wide Web (WWW) developed as a hypertext and multimedia retrieval system for distributed information that has emerged as the most popular information

delivery platform on the Internet. The WWW offers the potential to make online catalogues accessible through the use of gateways.

The expansion of the WWW since early 1990 has been truly amazing. It appeared as a simple communication medium for scientists and researchers; its many and pervasive tentacles have spread deeply into various organizations and homes around the world. By the result, our OPAC has been automatically being converted to WEB OPAC.

Authority Control in Online Environment (Web OPAC)

The Web environment opens up new uses for authority records and adds new objectives to augment the traditional objectives. Sharing and searching of authority records through Internet has simplified the creation and maintenance of authority record and reduced the cost of cataloguing of local libraries. Authority record in web environment enabled users to access information in the language, script and form they prefer, and it work as a link to digital resources such as biographical dictionaries, abstracting and indexing services, telephone directory and other reference sources in the web. With the help of authority records, new integrated library system displays reference information to direct users to authorized form of headings. Web catalogue and associated integrated library system provide the traditional authority control functions of creating and updating authority records and displaying the cross references, but have primarily been seen as a tool for cataloguers.

Cataloguers and others can use the authority file as another reference tool for name variations and information to identify entities and also as a channel for reaching bibliographic record and from there reaching directly to digitized resources. The records in this automated file also enable navigation to related entities.

Transliterated Names and Authority control

The name authority file links variant forms of a heading to the preferred form of heading to help preparation of cross- references. An author may use different names, or variant spellings may be used for his name due to transliteration process. Transliteration is

the process of formulating a representation of word in one language using the alphabet of another language. The aim of transliteration is to represent the script of a source language by using the letters or symbols of another script, usually in accordance with the orthographical conventions of the target language. By the result, a unique name in one language or culture may have variant spelling in another language. This is a situation similar to that of an author using different names or pseudonyms. In such a situation, the cataloguer should create cross-references from variant spellings (different name) to the preferred heading.

Transliteration has wider applications in daily life. Some are given below:

1. *Electronic representation* of non-Roman languages on platforms where Operation System support for such languages is not available. For example, texts in such languages as Arabic, Russian or Japanese can easily be stored in a multilingual database and manipulated as if they were a language like English,

2. Entering texts in non-Roman languages using an ordinary keyboard with a system that only supports the American Standard Code for Information Interchange (ASCII) character set,

3. Enables users unfamiliar with the non-Roman alphabet to read texts in that language by converting into the Roman alphabet,

4. Phonemic transcription used for pedagogical purposes, called *petrography,* enables language learners to study non-Roman languages in the Roman script, and

5. Cross-Language Information Retrieval (CLIR) of proper nouns and other information extraction operations can be performed by entering the transliterated string, which is converted before the search is performed. For example, entering *bn ldn* (transliteration)or *bin ladin* (transcription) to search for instances of *Bin Ladin* in Arabic.

Arabic/English International Transliteration Schemes

For consistency in transliteration process, the ideal solution would be to have a standard, internationally agreed, system. Several agencies have been proposed, but unfortunately none has been universally accepted. Probably the earliest attempt at standardisation was *Deutsche Morgenländische Gesellschaft* proposal, adopted by the International Convention of Orientalist Scholars in 1936. It is the system used in the *Hans Wehr Arabic Dictionary.* Another standard was agreed in 1971 at a conference of Arab experts in Beirut; and theoretically, at least, accepted by the countries of the Arab League. It has met some resistance, particularly in those Arab countries where French predominates over English. Another remarkable attempt was done by the ALA-LC. The Romanisation Tables adopted by the US Library of Congress and the American Library Association for cataloguing books has found its way into wider academic use. It covers a multitude of languages: there are 54 Romanisation tables for more than 150 languages and dialects written in non-Roman scripts.

However, in practice, the application / usage of international transliteration schemes is totally ignored, or found inconvenient in naming process. Names and naming activities are central to human symbolic and communicative process. To be human is to name and be named. The names of persons have shaped its form by the influence of culture, religion and the society. In the case of Muslims, they use Arabic names, even though they are living outside the Arabian countries. They use regional language to write their name, where the first transliteration process is done. Later, again they transliterate their names into English, for some official purpose. For example, a person born in Muslim family in Kerala possessing Arabic name, at first use Malayalam to write his name, and later transliterate it into English for any official purpose International/national transliteration schemes have no significance in this transliteration process and the problem of variant spelling for a unique name still continues here.

Transliterated Names and Unicode

Since the advent of Unicode, electronic representation of non-Roman characters is not an issue. As a universal character code set, Unicode provides a unique number to every character used in modern scripts throughout the world. The Unicode Standard is the

universal character-encoding standard used for representation of text for computer processing. Versions of the Unicode Standard are fully compatible and synchronized with the corresponding versions of International Standard ISO/IEC 10646. For example, Unicode 4.0 contains all the same characters and encoding points as ISO/IEC 10646:2003. The Unicode Standard provides additional information about the characters and their use. Any implementation, that is conformant to Unicode, is also conformant to ISO/IEC 10646.

Unicode provides a consistent way of encoding multilingual plain text and brings order to a chaotic state of affairs that has made it difficult to exchange text files internationally. Computer users who deal with multilingual text—business people, linguists, researchers, scientists, and others—will find that the Unicode standard greatly simplifies their work. Mathematicians and technicians, who regularly use mathematical symbols and other technical characters, will also find the Unicode Standard valuable.

The design of Unicode is based on the simplicity and consistency of ASCII, but goes far beyond ASCII's limited ability to encode only the Latin alphabet. The Unicode standard provides the capacity to encode all of the characters used for the written languages of the world. To keep character coding simple and efficient, the Unicode standard assigns each character a unique numeric value and name.

The Unicode standard and ISO/IEC 10646 support three encoding forms that use a common list of characters. These encoding forms allow for encoding as many as a million characters. This is sufficient for all known character encoding requirements including full coverage of all historic scripts of the world as well as common notational systems.

The Unicode standard specifies an algorithm for the presentation of text with bi-directional behavior, for example, Arabic and English. Characters are stored in logical order. The Unicode standard includes characters to specify changes in direction when scripts of different directionality are mixed. For all scripts, Unicode text is in logical order within the memory representation corresponding to the order in which text is typed on the keyboard.[7]

The application of Unicode helps to reduce the relevance of transliteration up to an extent. However, transliteration is inevitable in a database, which is designed to meet the

needs of people all over the world. In a database having Arabic names entered in Arabic using Unicode cannot be of use to such people who have no knowledge of Arabic language. It means that Unicode can solve the problem of regional languages, but such a database can be of use only to a limited people who know that language. So transliteration of personal names from regional languages is essential to meet the need of international users.

Authority Control versus Access Control

The access control record is the next generation of the authority record. It may be called as "super authority record" because of the potential it contains for enriched information for indexing. Access control records can be linked both to bibliographic records, to collocate all manifestations of a work, and to other related access control records, to collocate related works. The basic concept behind the access control record is removing both the label and notion of "authority". While authority control record declare a heading as "authorized" form, access control record links all variations without declaring one heading as authorized form. Access control records allow users to choose their preferred form or name, or to have displayed a default heading. It allows for more flexibility in display. The concept of access control entirely contradicts the whole second part of AACR–2, which is devoted to painstaking rules for how to construct authorized forms of names and titles.

Barhhart[8] puts a hypothetical access control records and compare with authority record as follows.

```
06    010       n 50018452
07    040       DLC ‡c DLC
08    100 00    Guillaume, ‡c de Machaut, ‡d ca. 1300-1377
09    400 10    Machaut, Guillaume de, ‡d ca. 1300-1377
10    400 10    Machault, Guillaume de, ‡d ca. 1300-1377
11    400 10    De Machaut, Guillaume, ‡d ca. 1300-1377
12    400 10    De Machault, Guillaume, ‡d ca. 1300-1377
13    400 00    Guillaume de Machaut, ‡d d. 1377 ‡w nnaa
```

Figure 1.4: Format of Name Authority Record

```
06   010       n 50018452
07   040       DLC ‡c DLC
08   400 00    Guillaume, ‡c de Machaut
09   100 00    Machaut, Guillaume de
10   400 10    Machault, Guillaume de
11   400 10    De Machaut, Guillaume
12   400 10    De Machault, Guillaume
13   400 00    Guillaume de Machaut, ‡d d. 1377 ‡w nnaa
14   999       ca. 1300-1377
```

Figure 1.5: Format of Hypothetical Access Control Record

In the hypothetical access control records, the 100 field switched with 400 field and 1xx field would be considered as "default" display. The 1xx field might not then be same in every catalogue of the same bibliographic record. Each library can fix their "default" heading and non-default form can use as reference.

The idea behind the access control is that an entity can be known by more than one name. An individual is an entity but may be called by different names by different people at different times in life. In the international realm living persons have name representations in many languages and script. The possible solution is that, instead of selecting one form of name as heading, give chance to select one of the heading as default headings according to their interest (local convenience) and give non-default as reference.

As stated earlier, finding is one of the functions of authority control, whatever may be the form of catalogue. Taxonomist refers to different spellings, synonyms, homonyms hierarchies and so on. The idea is to record these somehow, and link them to authorized nomenclature. Cataloguers do the same thing when they provide references from variant or related forms of a name. Information scientists also addressed the same problem. In this connection, it would be very helpful to understand how the task of retrieving variant spellings or mis-spellings of a unique name /entity was solved in modern Information Retrieval Systems.

Name Searching and Information Retrieval

Information Retrieval is the process of finding some desired information in a store of information or database. It implies the concept of selectivity. Information recovery is not the same as information retrieval unless there has been a selection process. Name searching is one of the major activities in information retrieval. The name searching may be either from the text or from a database.

Database name matching technology has long been used in criminal investigations, counter-terrorism efforts, and in a wide variety of government processes such as Visa processing. In this process, a name is compared to names contained in one or more databases to determine whether there is a match. Sometimes this matching operation may be straightforward exact match, but often the process is more complicated. Two names may not match exactly for a wide variety of reasons and yet may refer to the same individual.

Name matching can be defined as the process of determining whether two names (input and database) string are instances of the same name. It is a component of entity matching, but is distinct from that larger task, which in many cases requires more information than a name alone.

The challenges of name matching are greatly increased when databases contain names from outside the Anglo-American context. In China or Korea, the surname comes first, before the given name. In some countries such as Indonesia, many people have only one name. In the Arab world, a name may have many components showing the bearers lineage, and one of these is a family name. In addition to this, there may be multiple standard systems for transliterating a name from a native script (e.g. Arabic, Chinese, Cyrillic) into the Roman alphabet, or individuals may take their own Roman spelling based on their own understanding of how it sounds.

Some approaches have been made to take care of variant spellings within the Information Retrieval System. In most commercial information retrieval systems, the user only has the possibility to mark his query with elaborated wildcards to find derived or

similar words. Stemming algorithms is another technique used in many systems. But this technique is not appropriate in the case of non-regular (Out of Vocabulary) words.

Out of Vocabulary (OOV) words are a common source of errors in Cross Lingual Information Retrieval (CLIR). Bilingual dictionaries are often limited in their coverage of named-entities, numbers, technical terms and acronyms. A significant proportion of OOV words are named entities and technical terms. Typical analysis finds around 50%of OOV words to be named entities. Variations in the English spelling of words of foreign origin may contribute to OOV errors. For example, it was found that 32 different English spellings are in use for the name of the Libyan leader Muammar Gaddafi.

Similarity measures are widely used to solve the problem of variant spellings. Most of non-linguistic similarity measures can be subdivided into three categories: (a) String similarity measures which includes *n*-gram matching; (b) similarity with respect to typing errors, which counts the insertion, deletion, substitution, permutation and transposition needed to transform one string into another (Edit Distance Measure), and (c) phonetic similarity, which compare the similarity of sounds.[9]

_n_-gram matching

The *n*-gram similarity technique is a language independent which compares the letters of words regardless of the language used. If two strings are compared with respect to their *n*-grams, the sets of *n*-gram will be calculated for both strings. Next, these sets will be compared, and the more *n*-grams occur in both the sets, the more similar the two strings are. *Uni*-gram (i.e. n=1), *Di*-gram (i.e. n=2) and *tri*-gram (i.e. n=3) are widely used *n*-gram methods. In *di*-gram, two nearby letters are grouped and similarities are compared. For example, consider the word RECEIVER and mis-spelled words RECIEVER, RECEIVOR, RECEEVER. Firstly change the word RECEIVER into *di*-gram i.e, grouping of nearby two letters. So we will get RE, EC, CE, EI, IV, VE, ER. Then compare these pairs with the *di*-gram of other mis-spelled words as follows:

Table1.1 Example for the Use of *die*gram Analysis

Word	*Die* gram							
RECEIVER	RE	RE	EC	EI	IV	VE	ER	
1) RECEIVER	RE	EC	CE	EI	IV	VE	ER	7/7
2) RECIEVER	RE	EC	CI	IE	EV	VE	ER	4/7
3) RECIEVOR	RE	EC	CI	IE	EV	VO	OR	3/7

From the above example, the first word match fully with our input word (7/7). The second word is more similar (4/7) than the third word (3/7). The similarity is calculated as follows:

Similarity coefficient of word 1 and word 2 = $N_1 \cap N_2 / N_1 \cup N_2$ *where* N_1 *and* N_2 *are the n-gram sets of two words to be compared.*

Edit-Distance String Similarity Measure

A simple form of string similarity measure is an Edit-Distance measure. This method measures the number of single characters needed to be inserted and deleted to transform one string to another. For two string *s* and *t* of length *m* and *n*, respectively, the minimal such edit-distance can be determined by computing *edit (m ,n)* with recurrence to relation with the function *d (a, b)* returns 0 if *a* and *b* are identical and 1 or more, otherwise. For example, the edit distance between *hords* and *lords* is 2.

There are also more sophisticated versions of edit-distance that consider moves operator on blocks of letters and allocation of different cost to each kind of transformation.

Phonetic Similarity

In phonetic similarity, the pronunciation similarity of two string is compared. Soundex and Phonix are the two important methods that are widely used in phonetic similarity. Soundexs' phonetic property is restricted to the collecting of similar sounding consonants into different classes. Since Soundex and Phonix algorithms have been developed for the English language, they have to be modified for other languages. This can be achieved by adapting character class or substitution rules, respectively. For mixed language databases, (e.g., literature database with author from different countries), the Soundex algorithms is suited better because of its simplicity.

Soundex algorithms works as follows:

1. Remove all vowels, the consonants 'H', 'W', and 'Y' and all duplicate consecutive characters. The first letter is always left unaltered.

2. Create the Soundex code by concatenating the first letter with the following three letters replaced by their numeric code according to table set for this purpose. The numeric code is given below.

Table 1.2 Soundex Substitution of Letters with Numbers

Soundex	Numerical Code
B F P Y	1
C G J K Q S X Z	2
D T	3
L	4
M N	5
R	6

As an example, 'BAYER' and its variants like 'BUYER', 'BYER' and BEYER can be mapped into same sounded code and we can fix the rank. First calculate the

numeric code based on above table and compare both strings. Based on this classification, each word in database can be assigned and compared with query string.

Phonix

Phonix is far more complex than soundex. While Soundex only removes vowels, some consonants, and duplicate letters and carries out the numerical substitution, the Phonix algorithm is more elaborate. It works as follows:

1. Perform the phonetic substitution i.e. replace certain letter groups by other letter groups;

2. Replace the first letter by 'V' if it is a vowel or consonant 'Y';

3. Strip the ending-sound from the word;

4. Remove all the vowels, the consonants 'H','W', 'Y' and all duplicate consecutive characters;

5. Create the Phonix code of the word without its ending-sound by replacing all but the first remaining letter by its numerical values according to Table developed for this purpose. The maximum length of a Phonix code is restricted to eight characters; and

6. Create the Phonix code of the ending sound by replacing every letter by its numerical value. The maximum length of a Phonix code for an ending-sound is restricted to eight characters. Now each word of the database can be assigned to one of the three ranks: (a) identical, (b) similar, or (c) unrelated.

Table 1.3 Phonix Substitution of Letters with Numbers

Phonix	Numerical Code
B P	1
C G J K Q	2
D T	3
L	4
M N	5
R	6
F V	7
S X Z	8

Information Retrieval and Authority Control

The quality of online databases is very much related to the quality of data in the database. There will be variant spelling, omission, deletion or typing errors for a unique entity. By the acceleration of electronic publishing, there is increase in number of online databases. The challenge is to aquare correct information from the databases. One aspect of improving data quality is detection of variant names for a unique entity and linking them to improve searching. This is similar to authority work. The technique of authority work has vided implication in the effectiveness of Information Retrieval in online databases.

When a user keyed in a known form of a heading, the system would follows internal linkage and displays the requested item even though the preferred form of heading might be quite different from the form entered. There should be direct linkage between the form of heading entered and record displayed. The authority control mechanism works as invisible mechanism so far as a user is concerned.

The Study

"AUTHORITY CONTROL OF ROMANISED ARABIC NAMES IN ONLINE PUBLIC ACCESS CATALOGUE"

Definition of Terms:

Authority Control: The consistent use and maintenance of the forms of names, subjects, uniform titles, etc. used as headings in a catalogue.

Romanised Arabic Names: Arabic names written by using Roman letters. Romanisation and Transliterations are simultaneously used terms; however, transliteration did not specify the target language.

Online Public Access Catalogue: It is an automated catalogue with subject, title and author search options directly available to patrons.

Objectives of the Study:

1. To study and analyse the present methods and functioning of Authority Control process.
2. To study and analyse the Name Authority File of Romanised Arabic names with the help of Library of Congress Name Authority File.
3. Re-structuring of Authority Control process in par with web environment.
4. To study the problems of spelling variations of Arabic names in Information Retrieval process.
5. To explore the possibility of the application of Authority Control (or Access Control) mechanism in Database Name Searching and Free Text Name Searching.
6. Automation of Authority Control of Romanised Arabic names.

Authority Files of Romanised Arabic Names

In the case of transliterated names authority file, the cross-references are mostly variant spellings of personal names. While transliterating names into Roman script, by many reasons, variant spellings are used to denote a unique name. The usage of variant spellings compels the cataloguer to create cross-references to the selected heading.

A study on authority files of transliterated Arabic names in Library of Congress Name Authority File reveals many interesting findings. Most of 4xx entries (cross-references) are variant spellings of unique Arabic names. Some examples are given below:

1. 100 10 a Bashier, Zakaria
   400 00 a Zakaria Bashier
   400 10 a Bash¯ir, Zakar¯iy¯a
   400 00 a Zakar¯iy¯a Bash¯ir

2. 100 0 a Bas¯ir Totiyil
   400 1 a Totiyil, Bas¯ir
   400 0 a Basheer Thodiyil
   400 0 a Thodiyil, Basheer

3 100 1 a Basheer, Vaikom Muhammad, |d 1910-
   400 1 a Bas¯ir, Vaikkam Muhammada, |d 1910-
   400 0 a Vaikom Muhammad Basheer, |d 1910-
   400 0 a Vaikkam Muhammada Bas¯ir, |d 1910

4 100 1 a Bash¯ir Husain, Em., |d 1922-
   400 0 a Husain, Em. Bash¯ir, |d 1922-
   400 1 a Basheer Hussain, M., |d 1922
   400 1 a Hussain, M. Basheer, |d 1922

5 100 1 a Basheer, Tahseen
   400 0 a Tahseen Basheer
   400 1 a Bash¯ir, Tahs¯in
   400 0 a Tahs¯in Bash¯ir
   400 1 a Bashir, Tahseen

From the first analysis, it is very clear that the cross-references (4xx) are variant spellings of heading (1xx) which as shown below:

	Heading	**Cross-reference**
	(1xx)	*(4xx)*
1.	Bashier	Bashir
2.	Basir	Basheer
3	Basheer	Basir
4.	Bashir	Basheer
5.	Basheer	Bashir

A close analysis of headings (1xx) and Cross-reference (4xx) shows that some possible spellings are due to permutation of each other *i.e.* 'ie' as 'i', 'i' as 'ee' 'sh' as 's'. It is also seen that the heading (1xx) used in one authority file is the cross-reference (4xx) in another authority file and vice versa. This implication has wider importance in the automation of authority file of transliterated Arabic names.

Scope and Limitations

Arabic name includes four elements i.e., *Ism, Laqab, Kunya* and *Nisbah.* Among these, *Ism* is most prominent and it is an essential element of Arabic name, even though there are exceptions. The inclusion of other elements in Muslim names is differing from region to region. It would be place name as in `Basheer, Vaikom Muhammad;` or initial as in `Bash¯ir Husain, Em.;` or expansion of initial as in `Bas¯ir Totiyil`. But the *Ism*, the central elements, would be there in almost all Muslim names throughout the world. So the variant spellings or usage of other elements have less importance. This study is limited in *Ism.* In addition to this, the high computing power available now require minimum input in query, even though more elements in input would increase the precision of the search.

Conclusion

As stated earlier, the permutations of spelling in cross- references (4xx) and headings (1xx) have wider implication in the automation of name authority files. If we apply the techniques of access control in this context, it would give a better solution to the automation of transliterated Arabic names' authority file. In access control, instead of selecting one heading as preferred heading, all headings are linked to each other. Then, while the user searching on a heading would automatically be linked to the other headings. If we can create the variant spellings of an Arabic name through spelling permutation, we can link the non-preferred headings very easily. While the user is searching for one heading, the system automatically creates the possible variant spellings of the input heading and links them together. The burden of selecting one heading as preferred heading by showing the reference sources in the field of 670 in MARC records would be saved for the cataloguer.

REFERENCES

1. Ranganathan,S.R. *Classified Catalogue Code with Additional Rules for Dictionary Catalogue Code.* 5th ed. Bombay:Asia, 1964.155.

2. Cutter C.A. *Rules for a Dictionary Catalogue.* 4th ed. Washington: Government Printing Office, 1904.12.

3. International Conference on Cataloguing Principles. *Report.* Paris :ICCP,1961

4. Burger, Robert H.. *Authority Work : the Creation, Use, Maintenance and Evaluation of Authority Records and Files.* Littleton, Colo.:Libraries Unlimited,1985

5 ALA Glossary of Library and Information Science, Chicago, ALA, 1985.

6. Jabraj, V.F.D. "Online Public Access Catalogue" *Indian Journal of Information Library and Society* (2003) :147.

7 Unicode Home Page. 2004.26 November 2004 <http://www.unicode.org>

8 Barnhart, Linda ."Access Control Records : precepts and Challenges " 10 March. 2004 *Proceeding of Authority Control in 21st Century :An invitational Conference* <http://digitalarchive.oclc.org/>

9. Pfeifer, Olrich., Poersch, Thomas and Fuhr, Norbert " Retrieval Effectiveness of Proper Name Search Methods " *Information Processing and Management* 32.6 (1996) 667-679.

Chapter 2

ARABIC NAMING PRACTICES

Names and naming activities are central to human symbolic and communicative processes. To be human is to name and be named, and there by the possess full being and the ability to relate to the world in meaningful way. Traditionally, Arabic name has several elements. They are given below.

1. An *ism* (pronounced IZM, as the final syllable in the word *dogmatism*), a personal, proper name given shortly after birth, usually on the third day, but sometimes on day of birth and sometimes on the seventh day after birth. Examples of such names are *Muhammad* [Mohammed], *Musa* [Moses], *Ibrahim* [Abraham], *Ahmad.* Adults are seldom called by these names; socially it is considered a slight to address or refer to an elder or parent by their *ism.*

2. A *kunya* (pronounced COON-yah), an honorific name or surname, as the father or mother of someone; e.g., *abu Da'ud* [the father of David], *umm Salim* [the mother of Salim]. It is meant as a prefix of respect or reverence. Married persons (especially married ladies) are, as a general rule, simply called by their *kunya* (*abu* or *umm* + the name of their first son). When using a person's full name, the *kunya* precedes the personal name: *Abu Yusuf Hasan* [the father of Joseph, Hasan], *Umm Ja'far Aminah* [the mother of Ja'far, Aminah].

3 By a *nasab* (pronounced NAH-sahb), a pedigree, as the son or daughter of someone; *e.g.*, *ibn 'Umar* [the son of Omar], *bint `Abbas* [the daughter of Abbas]. The *nasab* follows the *ism* in usage: *Hasan ibn Faraj* [Hasan the son of Faraj], *Sumayya bint Khubbat* [Sumayya the daughter of Khubbat]. Many historical personages are more familiar to us by their *nasab* than by their *ism*: e.g., the historian *ibn Khaldun*, the traveler *ibn Battuta*, and the philosopher *ibn Sina* [Avicenna].

*Nasab*s may be extended for several generations, as may be noted in *ibn al-Haddad.* some of the examples set forth below. However, the vast majority of *nasab*s found in period sources are only one or two generations long. It is uncommon to find a

nasab which extends three generations back (considering the father of the individual as the first generation), and there are a very few examples which extend to four generations, such as *Abu Bakr Muhammad ibn Ahmad ibn Muhammad ibn Ja'far*

When the parent in a *nasab* is referred to by his *kunya*, the word *abu* becomes *abi*, *e.g.*, Muhammad's son-in-law was *'Ali ibn Abi Talib*, 'Ali the son of Abu Talib, or 'Ali, the son of the father of Talib.

4. A *laqab* (pronounced LAH-kahb), a combination of words into a byname or epithet, usually religious, relating to nature, a descriptive, or of some admirable quality the person had (or would like to have); e.g., *al-Rashid* [the Rightly-guided], *al-Fadl* [the Prominent]. *Laqab*s follow the *ism*: *Harun al-Rashid* [Aaron the Rightly-guided]

One particular form of *laqab* is formed on the pattern of *`Abd* [servant of] plus one of the 99 names of Allah; e.g., *'Abd Allah* (*`Abdullah*) [the servant of God], *`Abd al-Aziz* [servant of the Almighty], *`Abd al-Rahman* [servant of the Merciful]. These *laqab*s are used as, and in the place of, an *ism*: *'Abd al-Mun'im ibn Idris ibn Sinan.* The feminine form of this type of *laqab* is *Amat al-X*, for example, *Amat Allah* (*Amatullah*), (female) servant of Allah.

Sometimes what appear to be regularly-formed *laqab*s are found used instead of, or in the place of, an *ism*, *e.g.*, *al-Dahhak ibn 'Ajlan*, *Abu Talib al-Mufaddal ibn Salamah.* (Such *laqab*s might also be found used in the "normal" fashion for a *laqab*: *Muhammad ibn Ya'la al-Dabbi al-Mufaddal*.) There was no general rule by which *laqab*s are used in the place of an *ism*; the only reliable guide for proper usage right now is to look at actual period examples.

Some names are found both as an *ism* and in a *laqab* form: *Rashid* and *al-Rashid*, *Hasan* and *al-Hasan*, *Anbar* and *al-Anbar*, *Fadl* and *al-Fadl*. This is not, however, a general rule; not all *ism*s may be modified in this way and used as *laqab*s. Again, the only reliable guide for proper usage is to look at period examples.

5. A *nisba* (pronounced NISS-bah), a byname. *Nisbas* follow the *ism* or, if the name contains a *nasab* (of however many generations), generally follow the *nasab*. The three primary types of *nisba* are:

a) Occupational, derived from an person's trade or profession; *e.g.*, *Muhammad al-Hallaj* [Muhammad, the dresser of cotton].

b) Of descent, derived from the name of a person's tribe of birth or family lineage: *Mughirah al-Ju'fi* [Mughirah of the tribe of Ju'fi]; *Yusuf al-Ayyubi* [Joseph the Ayyubid, Joseph of the family line of Ayyub].

c) Geographical, derived from the place of residence or birth: *Yaqub al-Dimashqi* [Jacob of Damascus]. As is the case with *nasabs*, some persons in history are known to us primarily by their *nisba*: *Muhammad ibn Isma'il al-Bukhari*, the author of an early collection of *hadith* (sayings of the Prophet Muhammad) is better known from his place of birth, Bukhara, simply as *al-Bukhari*

These are the important elements that found in Arabic names. The practice of using these elements in naming process would not be same in all part of the world. However, the *ism* would be there in all most all Arabic names.

Reference

1. *Encyclopedia of Islam* Ed. Evan Donzel et al. New ed. Vol. 4. Leiden: E.J.Brill, 1978. 179-181.

Chapter 3

REVIEW OF RELATED LITERTURE

Review of related literature is a significant aspect of any research work. For any worthwhile study in any field of knowledge, the research worker needs an adequate familiarity with the work, which has already been done in the area of his choice. It provides information about previous investigations and procedure of previous researches. The reviews are grouped as sub-themes like OPAC, MARC, XML, AACR and Electronic Resources, Bilingual Catalogues, Regional Languages, Transliteration, Personal Names and Information Retrieval, Personal Names and Spelling Variations, Authority Control, Authority Control and its Cost, Authority Control Automation and Access Control.

OPAC

Behsesti[1] (1997) in the article "The Evolving OPAC" traces the development of Online Public Access Catalogue (OPAC). Advances in computer and communication technology have had an impact on online public access catalogue. The first generation of online catalogue had limited capability. Second generation OPACs were characterized by Boolean searching, proximity operators, keyword access, shelf list scanning, different display options and help facilities. However, it is limited to text-based command driven interfaces. Later, with the advances in microprocessors technology and introduction of powerful microcomputers, the distribution of tasks between the host/server and client became available. The Local Area Network allowed to develop client /server models in work place. The development of standard communication protocol like TCP/IP between server and client came into existence. The Graphical User Interface intensive application with a great variety of client platform ranging different versions of Microsoft Windows to Unix changed the face of OPACs. Advances in communication technology and implementation of standards have resulted in the exponential growth of Internet. The WWW was designed as the next generation of navigational aids for Internet, surpassing the text based systems and introducing hyperlink graphical interfaces in client/server environment. The client /server model, OSI, TCP/IP and web have allowed many libraries

to provide access to their OPAC through Web. Since the web was conceived as a hyperlink navigational tool, most webbed interfaces have browsing features. According to Behesti, librarians have to broaden their knowledge base more than ever before since webbed catalogues provide access to verity of remote resources. Librarians should be able to evaluate the content, scope and coverage of different information. Library and information professionals should be willing to use new ideas in creating next generation of OPACs, thereby presenting the users with genuinely intuitive systems.

Sridhar[2] (2004) conducted a comparative study of user behavior on the use of OPAC and card catalogue highlighting the differences in use of the same users and same library. The study was conducted at ISRO Satellite Centre Library in 1985 (card catalogue) and in 2002(OPAC). The study is based on observation techniques. The card catalogue has 25,000 documents (in 1985) and OPAC has 2 lacks records (in 2002). Also, the number of users doubled over the last 17 years (1985-2002). The study shows that OPAC terminal is used maximum on Wednesdays and minimum on Fridays. This confirms the findings of the use of card catalogue. The variations in the use OPAC during a day shows that OPAC use increases in middle of the days but use in afternoon is quite nominal. But the card catalogue was used more in afternoon. The OPAC is highly used to search database of books/documents (61%), journals (4%) and circulation/lending status (32%). Among the documents, 88% searches are executed on database of books. Among the access points user prefer while searching, title score high (38.3%) in OPAC and subject (54.2%) in card catalogue. Author access is preferred more in card environment (35.4%).

MARC, XML

Krieger[3] (1996) in the study "Characteristics of the 670 Field in Records for Names in the Anglo-American Authority File" seeks to gain further insight into the nature and use of the 670 field (source of See Reference) in MARC. He analyzed the field 670 to know that: (a) to what extent do name authority record rely solely on the piece in hand of information to justify their name headings, (b) to what extent do Name Authority Record (NAR) rely on other sources to justify their headings, (c) which types of headings use the highest number of NARs and (d) what is the average number of 670s per authority record in different categories. He also studied how frequently are electronic databases used in

citing pertinent information in a 670 fieled. For conducting the study, he took all the records coming under a broad subject 'Catholic Church' from Anglo-American Authority File via OCLC. He grouped the study as: (a) general discussion of the 670 field contained within the NARs of several different types of headings, (b) reference sources cited in the 670 field and (c) types of data in 670 field. The findings indicate that overall 42.13% of the NARs examined were employed a single 670 field citing the work being cataloged. Fifty three per cent of NARs for modern names, which form a large percentage of headings in current cataloguing, rely on the one 670; and for those that do not, only 13.89% cite a general reference or database. Records of this type could easily be created by most libraries or perhaps by a refined machine operation. According to Krieger, the increasing availability of electronic databases, which can be used in establishing headings, is important.

Johnson[4] (2001) in the article "XML and MARC: Which is "Right"?" explores the recent discussions on whether the MARC21 format would continue to be useful or XML (eXtensible Markup Language) replace the MARC. In the realm of World Wide Web, standards closely related to web development are essential. But MARAC21 format has many limitations: (a) information recorded in variety of ways in MARC, (b) there are inherent complexities in the format that are handled inconsistently across the format, and (c) MARC format is seen as "flat, with limited support for hierarchy." Equally, according to Lane Library Staff, AACR2 has too much emphasis on physical description and not nearly enough emphasis on access to intellectual content. Today, there is no need to worry about consistency of access points and regularizing the order of headings since keyword searching is familiar. On other side, XML has many advantages. It is a universal web data format and it employs Unicode to represent all data. According to Johnson, certainly MARC is not dead but absolutely XML has a future in managing digital library.

Fiander[5] (2001) in the article "Applying XML to the Bibliographic Description" puts three possible approaches to develop a Document Type Definition (DTD) for bibliographic data, each with different starting point. The first approach is begins with the MARC format and transliterate it into XML format by preserving the structure of MARC exactly. The second method is begins with the structure of AACR-2 and describe it via XML. In this method, a minimum AACR based DTD would replace the standard

punctuations like ". –" with XML tags, but further tagging and tag attributes would allow for the automation of much of what currently requires separate fields. The third approach is to take advantages of change in technology and incorporate some recent research into the area of cataloguing into the underlying descriptive codes before creating an XML structure. According to Fiander, the rigidity and internal irregularities of MARC are beginning to cause problems for cataloguers and users, and MARC is beginning to lag behind current research into bibliographic description standards. Rather than trying to patch up MARC, perhaps it's the time to start looking for alternative data format that provides flexibility for the next forty years. However, before libraries can migrate away from MARC into newer format, much further research into embedding current thought into practical data structure is required.

. Mansor[6] (2003) conducted a study on variant bibliographic record preparation standards in Malaysian libraries. A total of 410 similar MARC records were identified in three Malaysian University libraries. All the records were compared and differences were grouped in to three categories i.e. (a) format differences, (b) content differences, and (c) editing and inputting differences. The study shows that out of 410 records, only 83 records were exactly alike in terms of name headings. Three hundred and twenty seven records shows variations. The highest number of differences was found in content (279 records). It includes omission of whole field, different content, omission of part of entry and other differences. The study shows that, in the name field, honorific titles; date; and filial indicators are found as omitted. The variant ways of entering personal names were not only confined to local names but also in other name origins such as Western and Indonesian names. The author suggests that authority control of author's names should be done when integrating MARC records from different libraries in a single union database.

AACR and Electronic Resources

Banerjee[7] (1997) discusses the problems of describing remote electronic resources in the online catalogue. According to him, the bibliographic description of electronic resources is different from other information sources because: (a) it describes interactive documents, which are inherently difficult; (b) electronic resources are unstable; and (c) relationship between the library catalogue and electronic resources is different than that between the catalogue and physical materials. The catalogues and full text databases do

serve different functions, but every record in an online database must faithfully represent the document it describes. Even if some traditional cataloguing practices are invaluable for providing access to electronic resources, it is necessary to implement new technical solutions to provide adequate access to dynamic resources. According to Banerjee, cataloguing electronic documents consists of three basic principles i.e. (a) providing descriptive information which identifies the work, (b) determining access points needed for retrieval of catalogue record, and (c) recording the means by which the work itself can be accessed. Metadata and programs, which perform textual analysis, represent the most promising additions to cataloger's arsenal. If librarian wish to provide maximised access to the growing body of electronic information, it will eventually be necessary to develop new cataloguing techniques, which may involve shifting part of burden of collocating similar materials to creators and users of knowledge. Fortunately, advances in hardware and software technology are providing new tools to aid catalogers.

. In the article "Consequence of applying Cataloging Codes for Author Entries to the Spanish Library Online Catalogue" Ruiz-Perez[8] (2001) examines the consequence of the rules for choosing access points in online catalogue information retrieval systems in Spanish National Library Records based on Spanish Cataloguing Rules (1995) and some extent of Chapter 21 of the AACR-2. The researcher selected 838 records from the *Bibliografia National Espanola* database and analysed author access points. The procedure used to calculate the number and distribution of author access point did not distinguish between main and secondary headings, and allowed for all authors involved in a work to be used as entries. However, not all author names associated with a work are useful access points for retrieval, because not all persons were intellectually or materially responsible for the work. To find the author that are useful for purposes of retrieval, a system of categorisation developed based on the degree of responsibility of each author for the work. The 838 records were analysed for the variables: (a) source location of author access points, (b) potential access points, (c) used access points, and (d) unused access points. To assign author to location, he followed ISBD. For cataloguing potential access points, he checked MARC fields such as 245$c, 260$b which is not considered as access points, but that contains author name. To calculate access points actually used, he counted MARC fields defined as author access points such as 101,110,111, 700 and 711. The study reveals that, regarding potential access points, many authors can be involved in modern

monographs and all of them are potential or can be used as access points. An average of 4.25 access point were found per record. The location of access points includes title page-57.3%, table of content 33.5% and other 9.1%. A total of 2125 potential authors were not used as access point, i.e., the loss of access point is 59.5%. A total of 960 authors named on the title page (30.23%) were not used as entries. In works with up to three authors, 24.8% of authors were not used as entry point. In works more than three authors, 75.2% of potential access points were unused. If the access point from the table of content and title page were used, the author index would be more complete and accurate. The finding supports the urgent revision of catalogue rules for choosing author access point to make them more flexible, more practical and more in line with actual responsibility functions and type of authorship.

Bilingual Catalogues

Since the advent of Unicode as an international standard in 1993, the previously unchallenged believes in the impossibility of ever overcoming the natural division of non-Roman and Roman script expect thorough the process of transliteration. By this, a new breed of bilingual, bi-script catalogue is emerging, especially in Middle East. With an increase in the number of OPACs, which are available via Web gateways, the existing cataloguing rules and practices are inadequate to address some of the issues arising from creation, maintenance and use of merged bilingual, bi-script databases related to practically all aspects of bibliography ranging from descriptive cataloguing to authority control. Vassie[9] (2000) describes these issues and puts some solutions. According to him, four types of bibliographic databases exist in Middle East. These are: (a) monolingual, mono script catalogue i.e. all bibliographical descriptions and language depended qualifiers are in English, (b) twin lingual mono script catalogues i.e. Arabic script record held in a separate file from those in Latin script, using either Arabic or English as relevant for any notes, (c) monolingual bi or multi script catalogue which comprises records with notes in English and data primarily in Latin script, but with ability to enhance romanised records by adding data in an increasing number of vernacular script, and (d) bilingual, bi-script catalogue comprising Arabic language records and Latin script English language records merged in a single file. Vassie analyses the problems of multi script authority file

in bilingual, bi-script databases. According to him, two options exist for tackling authority control for names of authors represented in both available scripts in a bilingual, bi-script catalogue. The first option is to establish two headings for such authors in paired records. One is each script, with "See also" cross reference acting as link between them, and thereafter grouping bibliographic records under one or the other heading according to script of the descriptive fields, with additional "See" cross references in either script as required. The second options is to establish a single heading using the script of whichever language the author is normally associated with, and with "See" cross-reference from any other non-used form in either script. The first option reflects the policy established by Library of Congress, which has, however never implemented. However, the first option is to be justified to countries like Canada with more than one official language. The second option is the result of analogy with AACR-2. It supports favorable language of literary warrant. In the case of Middle East libraries, it seen relatively straight forward to create a rule ascribing literary warrant for catalogue purpose to any published usage in the dominant Arabic script, which would always take precedence over the Latin script.

The use of bilingual authority file in academic and research libraries has been successful in most of the multi cultural societies. In the study "Perception of Cataloguers and End User Towards Bilingual Authority File" Abduoulay[10] (2002) analysed and described bilingual authority file of the main library of International Islamic University of Malaysia (IIUM). The researcher also investigated perception of cataloguers and end-users in relation to the bilingual authority file. He conducted an interview with cataloguing staff. The IIUM main library authority file is organised according to AACR-2 principles. He analysed 51 records drawn from OPAC of IIUM and LC BP class (religious subject). The findings show that 48 records (96%) were found with personal authors. The headings were found in Arabic/English, if the work is in Arabic/English. The analysis shows that name and title authority file is provided in Arabic and Roman script, while subject authority file is given in English. The findings also show that the authority file can be searched through alphabetical or key word options. The researcher interviewed three cataloguing staff. The result shows that current tools used in cataloguing Arabic material are very useful. A majority of cataloguing staff feels the need for reference that will guide them in cataloguing Arabic materials. In addition, the majority of cataloguing staff believe that AACR2 and LCSH should be translated into Arabic. They ignore some of the rules of

AACR2/ LCSH in cataloguing Arabic items. There are many Arabic and Islamic terms that could not be found in LCSH. The researcher interviewed 23 end-users randomly selected. Among 23, fifteen were bachelors, five PG students and three research scholars. Out of 23, seventeen students used OPAC frequently. They prefer title as access point (39%), author (31%) subject (22%)and others 8%. All respondents believe that bilingual OPAC in the IIUM is beneficial. Fifty six per cent of students believe that more subject headings should be included and effective training or user education is necessary for the betterment of bilingual OPAC use.

Regional Languages

Olson[11] (1996) highlights some of the difficulties associated with the cataloguing of Thai materials and how it should be solved. Thai language is tonal and monosyllabic, each word being independent unit in a sentence. Romanisation of Thai language was started around 17th century. Later in 1968, Thai Royal Institute developed a romanisation table and LC/ALA also published a Thai transliteration table. However, romanisation of Thai languages created many problems. In Thai language, the tones are very important markers used to distinguish meaning of the words. Two words may be written exactly the same but have different tones and, hence, they have different meaning. The LA/ALA Romanisation system is done according to pronunciation of particular syllable and it is found that two words with similar pronunciation create confusion. Thai language has no space between words and no punctuation marks. Therefore, dividing words for inputting and retrieving bibliographic records is a task. The rules formulated by LC/ALA for dividing words were more complicated. Variation in pronunciation is another dilemma. Some Thai words can be pronounced in more than one way. Thai personal names are presented in direct order. According to Olson, Thai Name Authority File has to be created on the basis of the LC/ALA Romanisation Table. It would be advisable to LC and ALA to establish a committee of cataloging librarians fluent in Thai to work out an improved and theoretically viable system of romanisation, so that this problem is not carried over too far into the next millennium.

Chinese people, the biggest human mean on the globe, have spread all over the world in last couple of hundred years. The Chinese language is very different from

alphabetical languages. Yewang Wang[12] (2000) in the article "A look into Chinese Persons' Names in Bibliography Practice" describes the transliteration of Chinese personal names. Transliteration of Chinese persons' Chinese name merely represents the Chinese name in Roman script. There is no direct connection between the transliteration and person. There are as many as more than ten forms of Chinese persons' name as transliterated or real in non-Chinese language. For making these forms easily distinguishable, Yewang Wang developed *'sheep-fox method'* in which persons' original name is the *Original sheep* and its transliteration represent a *Surrogate Sheep*, but if transliterated name is used as real name in another language, then there would not be *surrogate sheep* instead new real name of person's original name become *fox*. According to Yewang Wang, Chinese persons' name in Roman script language can be divided into five categories. In the first category, Chinese persons' name in Roman script language is exactly the same as the standard transliteration of the persons' original Chinese names. The second category includes Chinese persons' name in Roman script languages initially transliterated from their Chinese names according to Mandarin pronunciation, but in a non-standard form. The third category is Chinese persons' name in Roman script languages initially transliterated according to dialects or unusual pronunciations of the original Chinese names spoken. In the fourth category, Chinese persons' names are in Roman script languages and the given name or surname or full name is typically European. The fifth category includes the name of some special "Chinese" who are not Chinese by birth, but have lived or done activities in/with Chinese world. They have names in Chinese that have been used practically. A typical Chinese persons' name in Chinese is with surname format. If a transliterated form is merely a transliteration representing the original Chinese name in non-Chinese languages, it is likely to retain that order. Most Chinese surnames have just one character while some have two. Chinese overseas having names in Western languages generally reverse the surname to the rear according to western customs. But in South Asia, the order remains same. Yewang Wang further describes the romanisation policy of Chinese names. If a person's Chinese name is clearly most commonly known, or appears most frequently in the person's work, or reference sources, his or her name heading in our databases will still be the standard romanisation of his or her Chinese name. On the other hand, if a Chinese has a name in a non-Roman script language like Russian, that is clearly most commonly known or appears

most frequently in the persons works or reference sources, then his or her name heading will be the romanisation of his or her Russian name. He concludes that the name authority file records of Chinese persons' need enhanced notes to indicate clearly whether a heading is a *Sheep fox* or *surrogate sheep*. If the list of sources used for the name's establishment cannot effectively do so. Neutral *sheep fox* and Dark *Sheep fox* need such a note when their appearances are similar to standard sheep. Alternative names of persons in other script than his or her preferred name heading should always be made non-preferred headings.

Khurshid[13] (2002) reviews major problems in Arabic cataloguing and try to answer the questions as to why not yet been able to solve it. According to him, the major problem is arising from lack of bibliographic information such as imprint, date of publication and edition statement. Even though modern Arabic publishers generally follow the patterns of Western publishers, a small percentage (5%) of books still have the problems of the early books, in which, an author's name and imprint information given elsewhere in the book. It takes time for a cataloguer to identify those elements and prepare a description of the book. Some of the Arabic publications do not have date of publication at all, while others have date given in odd places. Some publishers give date in the Gregorian calendar while some follows Hijia calendar. The edition statement is mixed with impression. The patterns of Arabic names also need special attention. Arabic names are generally divided as classical and modern names by taking the year 1800 AD as the beginning of modern period. *Ism, Nasab, Nisbah, Kunya, Laqab* and *Khitab* are major elements of classical name. Determining the entry elements of classical name is not easy. The Rule 22.22B of AACR-2 requires several cross-references from the best-known element which is taken as entry element. Unlike classical names, modern names are composed of fewer elements including *Ism, Laqab* and *Kunya* . As in the case of classical name, the entry element in modern name is also best-known element, which is normally the last element of the name or surname. According to Khurshid, it is preferred to compile an authoritative list of both classical and modern names on the line of Library of Congress Name Authority File and use it consistently. The other problem faced by Arabic cataloger is subject heading. The short coming of international classification system and subject headings list in dealing with subject cataloging requirements of Arabic material have forced cataloguer to find a local solution. The Sear's List of subject Headings and Library of Congress Subject

Headings are biased to Western culture. Therefore, Arabic librarian felt the need of prepare their own subject headings list based on the characteristics and grammar of Arabic language and culture of Arabic world. Like Library of Congress Subject Headings and Sear's List, the classification schemes such as DDC and UDC are also biased to western culture. The coverage of Islam and Arabic literature, history, religion is very superficial in these schemes. Unlike the Christianity with 70 numbers (220-280), Islam is placed under other religions (290) and given the number 297. To classify thousands of books in Islam, Arab librarians are compelled to adopt local expansions, but without unification. The MARC format not fully supporting Arabic script creates serious incompatibility problems. Over two decades have passed while deliberating the issues of whether or not a different MARC format is required to Arabic materials like USMAR and UKMARC. The cataloging problems of Arabic materials are still continuing. According to Khurshid, it is due to the absence of initiative by the national and regional agencies willing to come forward to address the problem.

Plettner[14] (2003) in the article "Arabic Name Authority in the Online Environment: Options and Implications" discusses authority file format specifically MARC21 format and theoretical advances and efforts made by the different agencies. MARBI (the American Library Association joint committee responsible for development and maintenance of MARC format) has added capabilities for addition of alternative characters in MARC21 authority record. NACO (the National Name Authority Cooperative) has been involved in establishing Arabic names in LC authority records in the Name Authority File. British Library Name Authority File has decided to merge files with USNAF to make up the Anglo-American Authority File (AAAF) to reduce duplication and make it even more international. IFLA is a very important hub in the design of names donated by national biographical agencies. Authority record formats and styles that are developed should be compatible with the computer system that is designed to handle them. Any incorporation of new models has implications for the synthetic structure and indexing of the databases using them. Two models of multi-lingual and/or multi-script authority records have been proposed in Concise USMARC21 format for incorporation of original script in authority records. The author suggests that: (a) an International Standard Authority Data Number (ISADN) should be created, (b) the provision for easy exchange of MARC format should be developed, (c) reduce the unnecessary redundant subheadings

in MARC should be reduced, (d) the new format hould be easy to use and to share, and (e) a centralised clearing house of multi-lingual and multi-script name authority records will be created.

Transliteration

Arbabi[15] *et. al.*(1994) developed a hybrid algorithm that automates the transliterations process. The generation of Roman transliteration of Arabic names is accomplished in several stages. The name must, first be entered into a database (in Arabic), and then vowelized, since Roman text requires vowels. After vowelization, each name must, then be given a phonetic Roman representation as a base, from which the multiple Roman spellings are produced. Arabic name database can be created either by typing or using Optical Character Reader. Then, vowelize by short vowels into Arabic names. Using hybrid algorithms, comprising a knowledge-based system and Artificial Neural Network, can complete automation of transliteration process. The knowledge based system approach allows the linguistics to include all rules of vowelization that can be codified simply in declarative form. Artificial Neural Network provide the capability to develop and use statistical heuristics which are not know and/or cannot be represented as declarative rules. The proposed transliteration system has four parts i.e., main program, knowledge base interpreter, knowledge base and a set of utility routines. The main program is written in C language. Knowledge base is written in CLIPS and consists of two files loaded at run time by the main program and later jointly interpreted by CLIPS as on knowledge base. Artificial Neural Network is used to eliminate that name which cannot be properly vowelized using expert systems. The system developed by Arbabi, for IBM Federal System Company, is applicable to a wide variety of purposes including Visa processing and document processing by border patrols.

Stalls, Bonnie Glover and Kevin knight[16] (1998) conducted a study on back transliteration i.e. recover the original script from the transliterated word. Transliteration is more complicated process for language pairs that employ very different alphabets and sound system such as Japanese/ English and Arabic/English. Arabic has three vowels (a,i,u) each of which has short and long variant, plus two diphthongs (ay,aw), whereas English has a much larger inventory of as many as fifteen vowels and no long contrast. For back transliteration, the researcher developed a new statistical model that converts English

phoneme sequence directly into Arabic writing. They built 150-word bilingual dictionary for training data and applied EM learning algorithms to produce Arabic sound from English sound. Initial result was satisfactory. The program leaned to map English sound onto Arabic letter sequence. They applied three probabilistic models to obtain the top English back transliteration for each word. The back transliteration contains the following steps: (a) produce English sound sequence with the help of sound-letter mapping table, (b) produce new English Phrases with their probability, and (c) re-score and select these phrases as per the rank. The program offered many good translations, but have errors of omission and commission. English G is incorrectly produced from its voice-less counter part in Arabic K. The study shows that in the English- Arabic sound transliteration systems, the letter D, S and Z make problem.

Jeong[17] *et al.* (1999) conducted a study on foreign word detection in the Korean text and back-transliteration and analysed its importance in Information Retrieval. Automatic identification of foreign words was done with the help of statistical techniques that relies on phonic differences between foreign words and Korean words. Back transliteration is more challenging and of more practical interest. It highly depended upon the context in which the foreign word appears. In order to evaluate efficiency of back transliteration, the researcher used a training data set consisting of 1200 pairs of foreign words with the corresponding English words and a test data set consisting of 100 foreign words not included in the training data set. The training data set was constructed manually by segmenting of foreign word into consonants and vowels and associating each element with the corresponding English alphabet in the original English word. They measured how many of 100 foreign words are correctly converted into corresponding English words. The result shows 58% accuracy in training data and 47% accuracy in test data. Within the top 10, training data shows 96% accuracy while test data shows 93% accuracy.

Nasreen, AbdulJaleel and Leah S.Larkey[18] (2002) presents and evaluate a simple statistical technique for English to Arabic transliteration. The technique is free form heuristic or linguistic knowledge of either language. This technique is based on probability of Arabic alphabet with corresponding English alphabet. The probability is counted from a sample of English –Arabic transliterated source. For example, Arabic alphabet ز is transliterated as 'S' and 'Z' simultaneously. The probability of S and Z is calculated as the

number of occurrence of 'S' and 'Z' in the sample data. If there is total 10 ز in the sample and 'S' is used 6 times and 'Z' used 4 times in transliteration, the probability of S is $p(S/ز) = 0.6$ and probability of Z is $p(Z/ز) = 0.4$. The basic issue for such a statistical transliteration model is how to select the sample and how to use the model to generate new transliteration. One can generate a single transliteration by taking highest probability transliterations for each character, or by taking one transliteration for each character at random, according to the character transliteration probability. One can generate all possible transliterations mapping with non zero probability. However, the character-by-character transliteration will not be correct always. For example, the Arabic alphabet س is transliterated as 'S' and ح as 'H, but if 'H' follows 'S' i.e. 'SH' the correct Arabic alphabet is ش not ح س. Hence, a more sophisticated model would have probabilities for character pairs (*bi*gram) rather than single character (*mono*gram). The *bigram* method is more efficient in such a case where one character produce different pronunciations when it joins with other characters. Nasreen and Larkey tested the effectiveness of newly generated statistical model with a list of 148,599 English and Arabic proper nouns obtained from Arabic Proper Noun Dictionary. Three experimental conditions were compared with the help of dictionary, i.e., (a) monogram (character-by-character) un-aligned, (b) monogram aligned, and (c) *bi*gram aligned. The finding shows that aligned words give more accurate transliteration. Out of 5000 names of sample, 53.5% were correct in aligned monogram while 30.2% only in un-aligned monogram. The *bi*gram produced 70.3% correct transliteration, which is most effective method.

Out Of Vocabulary (OOV) words are a common source of errors in Cross Language Information Retrieval (CLIR). Bilingual dictionaries are often limited in their coverage of name entities, numbers, technical terms and acronyms. One way to deal with OOV words is to transliterate unknown words. Nasreen Abdul Jaleel and Leah S Larkey[19] (2003) conducted an experiment on OOV words and formulated a statistical technique to train an English to Arabic transliteration model from pairs of names. There is great variability in the Arabic rendering of foreign words, especially named entities. According to them, the transliteration source that generates multiple Arabic spellings would be useful for cross language information retrieval. A statistical transliteration model can be used to

generate many alternative spellings, and therefore, lends itself well to the task of generating transliteration for cross language information retrieval. The new model produces strings of Arabic characters from a string of English character. To generate Arabic transliterations, the English word is segmented, and for each segment, all possible transliterations are generated. The transliterated Arabic names are then, ranked by using the following formula:

$$P(w_a \; w_e, w_a \in A) = P(w_a \; w_e) * P(w_a \in A)$$

where $P(w_a \; w_e) = \prod P(a_i \; e_i)$

and $P(w_a \in A) = \prod P(a_i \; a_{i-1})$

A handcrafted model was developed to provide a higher quality benchmark for measuring the performance of the automatically trained transliteration model. The researchers selected 37,000 names from AP News articles in English. The corresponding Arabic transliterations were obtained from the online bilingual translation system such as NAMSU, *Almisbar* and *Tarjim*. The output of the transliteration models were evaluated in two different ways *i.e.*, measure of transliteration accuracy and the retrieval effectiveness of cross language information retrieval. The ranked spelling variations were grouped as 'first 1', 'first 5' and 'first 20' and matched with NAMSU, *Almisbar* and *Tarjim*. The result shows 68.9% accuracy for 'first1' variation, 84.8%for 'first 5' variation and 89.8% for 'fist 20' variation. It means that if more spelling variations are produced, the cross language information retrieval will perform well. According to researchers, the result of the experiment support the following generalizations: (a) good quality transliteration models can be generated automatically from reasonably small data sets, (b) a hand-crafted model perform slightly better than the automatically-trained model, (c) the quality of the source of training data affects the accuracy of the model, and (d) context dependency is important for the transliteration of English words and the result of the information retrieval evaluation confirm that transliteration can improve cross-language information retrieval.

Personal Names and Information Retrieval

Pfeifer, Poersch and Fuhr[20] (1996) in the article "Retrieval Effectiveness of Proper Name Search Methods" analyses the problems of miss- spelling, typing error and different transliterations of proper names in databases. Information scientists developed a number

of methods to solve these problems. One method is stemming algorithms, which reduce words to their *basic form* or *stem form*. This method is useful for regular words, but it is not appropriate for Proper noun searching. The other method is non-linguistic similarity measures that can be subdivided into three i.e., (a) string similarity, (b) similarity with respect to typing errors, and (c) phonetic similarity. The phonetic similarity method can again be subdivided into two i.e. Soundex and Phonix. The researcher conducted a study on retrieval effectiveness of Soundex, Phonix and combinations of similarity measures. Both in Soundex and Phonix methods, the strings are replaced by their numeric code prepared for this purpose and ranking them (i.e., miss-spellings, typing errors and variant spellings). Another well-known method is the statistical method using *n*-gram. It is language independent method. The researchers conducted a study on retrieval effectiveness of above said methods and compared with combinations of these methods. For experiments, surnames were manually extracted from a couple of sources and created a database with 18,000 words named as 'COMPLETE'. Then 90 names were randomly selected from this database. Recall-precision graph were used to compare the retrieval effectiveness. Because of the test queries taken from the database, the precision is 100%. The study shows that Soundex is the worst variant because of missing phonetics substitution. The *n*-gram method was analysed for n=1(unigram) to n=5 and n-1 blanks. Here n-1 blanks give best result. The increase in *n*, performance is decreasing. The *di-gram* (n=2) with one blank is the best variant of *n*-gram. *Tri-gram* (n=3) with two blanks is slightly worse. Much better than the use of single techniques, combination of these techniques shows better result. For this, the ranking produced by different methods were mapped onto a single linear scale. The study shows that the combination of *di-gram* with one additional blank and Phonix can be recommended as best method to solve the problem of miss-spellings, typing errors and variant spelling in databases.

Lewellen[21] (1999) conducted a study on the application of Artificial Neural Network (ANN) in name searching. ANN is viewed as a promising technology for name searching, since it is able to overcome the problems of variant spellings of personal names. Word representation by numerical value is the basic of application of ANN in name searching. It means that letters are symbols, when ANN values are numeric. Lewellen developed several word representations designed specifically for recognition of variant spellings of natural language words. It includes: (a) left FLLB (Fixed –Length Letter

Buffers), (b) split FLLB, (c) Bi digram FLLB, (d) Bi-gram CCSM (Coarse- Coded Symbol Memory), and (e) Bi/Unigram CCSM. The researcher conducted study on the following questions: (a) do different ANN word representation cause difference in capability of back propagation of ANNs to perform name searching, (b) which representation are most effective, (c) do the word representation have identifiable strength or weakness on specific type of spelling variations, and (d) can the result of all these studies be synthesised into useful generalisation about the influence of word representation or ANN performance. In order to answer these questions, he developed a back propagation network using five different representations method. The trained network, then, tested with corpora containing specific types of spelling variations. The test result shows that word representation significantly affected the ability to an ANN to recognize spelling variant of natural language words; it is not the case that an ANN is able to learn and generalise regardless of word representation type. Further, different word representations have identifiable strength and weakness for specific types of spelling variations. Relatively, holistic representations- the bi/unigram CCSM and split FLLB – improve ANN performance by reducing its dependence on letter position. ANNs employing such holistic representations are able to process the type of spelling variant encountered in name searching of Romanised Chinese names.

Larkey, Nasreen Abdul Jaleel and Margaret Connell[22] (2003) in their article "What is in a Name? Proper Names in Arabic Cross Language Information Retrieval" overview the role of variant kind of translation resources in a typical Cross Language Information Retrieval.(CLIR). Proper nouns are an important cause of Out Of Vocabulary (OOV) error in IR. The proper noun includes names of people, places, and organization and acronyms-nouns that are typically capitalized in English. Another problem with names in Arabic – English information retrieval is great variability of spellings. The study analyses how much performance degrades while using a bilingual lexicon that lacks proper names, and examines several different sources of proper name transliteration from English to Arabic and its effectiveness. All experiments were carried out using the TREC 2001 collection of 383,872 Arabic News paper articles form *Agence France Press* (AFP) and 50 TREC 2002 topics. Queries were formed from the title and description field of English version of the topics. Arabic articles in the collection were converted by using Unicode UTF-8 encoding into Windows Arabic. Simple tokenization broke up text into words while space or

punctuation characters are not found in English text. Arabic text was stemmed using light stammer. For cross language querying of Arabic collection, a dictionary based query translation method is used. Retrieval experiment contains the following steps: (a) each queries was tokenized, lower cased and stop words were removed; (b) each English word was looked up in the bilingual lexicon and all the synonyms were placed inside synonyms operators, then the translation or synonyms for all query words were gathered under a weighted sum operators. The weight on each translation (terms of synonyms) was the weight of English source word in expanded English query; (c) the structured Arabic query were expanded much like the English queries; and (d) the expanded Arabic query were submitted to AFP collections. The study showed following results: (a) Proper nouns are an extremely important component in cross language information retrieval. Mean average precision degrades more than 50% using the typical bilingual dictionary that does not include proper names, (b) Sources of proper names vary in quality at least for a language pair like English and Arabic, in which there is tremendous variation, both in how English-Arabic and Arabic- English rendering, and (c) a good strategy is to use transliteration for proper names or to use transliteration for unknown proper names. The researcher suggests that it is better to generate multiple alternative transliterations for unknown words rather than one. It is safe to include up to 20 alternative spelling variations for the unknown query word.

Spink and Jansen[23] (2004) conducted a study on how people search for information on other people using personal names via web search engines. They analysed what are the characteristics of personal name web searching and the retrieval effectiveness of different techniques in name searching. The researcher collected and analysed actual queries that searcher submitted to two commercial web search engines i.e. *AlltheWeb.com* and *Alta Vista.com* in 2002. They randomly selected 10,000 queries from 1.2 million data for each search engine using Poisson Sampling techniques. Firstly, they grouped queries into two groups *i.e.* personal name queries and non-personal name queries. Personal name queries are further classified as: (a) celebrity/non-celebrity names; (b) with/without double quotation in searching; and (c) with/without additional terms in searching. Again they randomly selected 100 queries from *Alta Vista* and submitted each one to four search engines *i.e.*, *AQL*, *Google*, *MSN* and *Yahoo*. The top ten result of each query analysed and estimated retrieval effectiveness by an independent evaluator. The same query again

resubmitted with double quote and analysed the effect of double quote in information retrieval. The study shows that only 4% queries were personal name queries. Out of this, 25% queries were related to celebrated names. Eleven per cent of personal name searches in *AlltheWeb.com* and 24% in *AltaVista* were submitted in double quotes. The researcher conducted *t* test to evaluate the effectiveness of double quotes in name searching. The result shows that double quoted personal name queries generally produced more specific retrieval. They also analysed the effectiveness of different search engines. *Google* retrieved high result (83%) and *MSN* (78%) shows the lowest. *Alta Vista* users use double quotes more. *AlltheWeb.com* had s slightly higher use of personal name queries. The study has wider implication to search engines designers. People also need to put more effort into improving their information behavior.

Personal Names and their Spelling Variations

Weintraub[24] (1990) in the article "Personal Name Variations: Implication for Authority Control in Computerised Catalogues" explored the effect of personal name variations on authority control and data retrieval in computerised catalogue by studying names of 395 persons receiving entries in the catalogue of University of California. The objective of the study was to provide a more complete picture of personal name characteristics in bibliographic and authority files. The minimum sample size was calculated to be as 383 by using formula $N= (z/e)^2 (p)(1-p)$,where 'e'= error level(taken 0.05) 'z' represent 95% confidence interval and 'p'= probability. A simple random sampling method was used to identify the sample of bibliographic records. Main and added entries in the records for all persons whose name appeared in title or statement of responsibility areas were selected for the study. All variations of 395 names in bibliographic and authority record were examined. Bibliographic records were identified from the University of California Library. The study shows following results: Of the 395 persons represented by headings sampled, 36.7% (145) have entries for only one title, 16% have two titles and 10.3% have three titles. Given large number of persons with entries for more than one work, 81.5%(322) appear in one form in all bibliographic transcriptions. Name variations in authority records are also low. A comparison of transcribed name forms to titles reveals that 44.48% (177) of all people whose names appear only in one form in bibliographic transcriptions received entries for more than one title. Furthermore,

the cross tabulation of transcribed names with controlled headings and references clearly shows that most names, as taken from title pages, match established forms in authority records. The most common type of variation among name forms is fullness of forenames (38.8%), added initials (27.2%), difference in forenames (13.6%), difference in surnames (6.8%), and other terms (6.8%).

Strunk[25] (1991) conducted an investigation on frequency of documents with *form problems* of personal names and the conditions, which constitute form problems in the registration of personal names and frequency of each type of form problem in personal names. The study was based on Danish version of AACR2. According to him, the *form problem* occurs when there is a variation in the form of the name in the description and the personal name heading prescribed by the Danish version of AACR2, or Danish version of AACR2 has prescribed a reference. The sample was taken from Danish National Bibliography. The sample was limited to books and yearbooks in last five years. Out of 167280 records, every 150th record (total 1116) has taken as sample and marked the records containing *form problem* (total 390). These 390 records contain 463 names. The distribution of types of problems in personal names and suggested solutions were: (a) names alone not sufficient for identification (57.67%) occurs amazingly often. No device in the online catalogue can automatically individualize names, which are not sufficient for identification, (b) compound surnames (31.53%) may more or less handled in online catalogue without traditional authority control,(c) variations in fullness (12.53%) can solve by truncation, (d) variations in Romanization (1,51%), there is no way to automatically bring the variant together and (e) change of name (1.08%), the various names of an author can be brought together by traditional authority control. According to Strunk, the expanding device of online catalogues may replace authority control, as listing of all names may provide an overview of all the forms of names in the catalogue.

Taylor[26] (1992) conducted comparative study to determine the extent to which OCLC bibliographic records contain variants in personal and corporate name access points with Library of Congress Name Authority File (LCNAF). The researcher analysed: (a) what proportion of records in the OCLC database has name access pints for which LCNAF records may be found? (b) How often do the name access points and the authorised forms agree exactly? (c) How often do the name access points agree with

references on the LCNAF records? (d) How often do name access points found on ample OCLC records? (e) In what ways do access points for a name vary from the standard form for that name? and (f) Are the variants in access points could be corrected by a computer program?. Sample size was fixed as 929 by using formula $N= (z/e)^2 (p)(1-p)$ and limited the sample as 450, for convenience. The study shows following results: Of the 450 sample records, 25 (5.6%) have no personal or corporate name access points. The remaining records (425) yield 457 personal names and 153 corporate names. Authority records were found for 275 personal names (out of 457) and 126 corporate names (out of 153). Two hundred ninety records have one or more names for which authority record was found in the LCNAF. Out of 290 records, 68 have one or more names that differ in form from the form on the corresponding authority records. The analysis of variant headings shows that near match to the standard as 42.9%, single typographical error as 29.4%, near match to reference as 11.7%, and exact match to reference as 5.2%. The variation in headings happens by: (a) omission/ inclusion of date of birth / death, (b) spelling, fullness of first forename, (c) initial difference, (d) forename entries spelled differently, (e) surname entries with comma missing and spelling, (f) different entry word and (g) punctuation, spacing, capitalization.

In the paper " Searching on Full Name Providing for Spelling Variations" Gideon de V. de Kock[27] (2002) discusses the handling of spelling variations of names, aliases and/or derivatives of names and double barrel names (hyphenated names). Various methods so called Soundex and weighted transformations ware used to solve these problems. Koch proposes new method. In the proposed method, names considered as equivalent be grouped together in equivalent classes and stored in a name database that is being updated on a regular basis. Double barrel names were put in the same equivalent class. Within each equivalent class, one name is identified as the "characteristic name" that represent all the names in the class. An equivalent name index was built. The major shortcoming of this method is that the allocation of a name to an equivalence class. Often equivalence class is too large and a full name will appear in more than one set. According to Kock, future works of new method includes the use of inverted indexes on the name using the similarity sets and investigate and evaluate some of the search techniques used on the Internet. Such method should fully provide for the permutation of forenames and partially for the dropping and/or addition of forenames.

Raghavan, Hema and James Allan[28] (2004) in the article "Proper Names and their Spelling Variations in Automated Speech Recognition Output" proposes new methods of normalising names in Automated Speech Recognized (ASR) documents. Edit distance methods is the common method that widely used to solve the problem of spelling variations in Information Retrieval. The researchers addressed the same problem by grouping names that "sound like" each other together, without the use of contextual cues. They used a clustering algorithm to determine whether any two words are linked or not. They generated pairs of linked words (names) in many different ways. One is using statistical machine translation method, which translate the words and group the Out Of Vocabulary (OOV) words (names). The second method is manual creation of pairs. From the two lists, they generated other five lists based on various parameters. They applied two methods for evaluation. One is *Intrinsie* (Paice) evaluation method, which measures the performance of a stammer based on its Understamming (missed form the group) and Overstamming (false in the group) indices. The second method is *Extrinsie* evaluation, which is based on precision and recall metrics. The name query experiment shows following results. The name 'Seigal' with equivalence class "Seigal', 'Segal','Siegal' 'Siegel " got 100% recall and precision. In the case of equivalence class "Lewenskey, Lewinski, Lewinsky" the term Lewenskey has a string edit distance of 2 shows lower precision. The equivalence class of "John-Jon-Joan" has very low precision and recall. Similarly, in the case of query class "Chiang , Ching " with edit distance of 1 resulted in a lowered precision. The generative methods are able to track certain kinds of variations in spelling due to similar sounding alphabet like 'i' and 'y', 'c' and 'k'. According to them, the new method will help to reduce computational expense than the Edit Distance method.

Toivonen[29] *et al.* (2005) presents a novel two-step fuzzy translation technique for cross-lingual spelling variations. Transliteration refers to phonetic translation across languages with different orthographic. In the proposed method, no phonetic elements are included. In the first step of the new technique, some language words are transformed into intermediate form by means of transformation rules. The intermediate forms are often correct transliteration, or similar word forms to their target language equivalents than the original source language words. A transformation rule refers to an automatically extracted regular correspondence between the characters in two languages. The transformation rules were generated automatically by extracting equivalent term pairs from transliteration

dictionaries. The forms were then aligned pair wise and regular correspondence were identified using the Edit Distance measure. As a next step, among all transformations that produced minimum edit distance, one transformation was selected on the basis of smallest sum of error values. From the selected transformations, the transformation rule was generated based on the frequency of occurrence of similar rule. In the second step, the intermediate term obtained in the first step is translated into a target language using *n*-gram matching similarity between the intermediate form and words in the target word list and compared to obtain a ranked list of target words. The two set fuzzy translation gives the following result: (a) the combined transformation rule based translation and fuzzy matching machine technique perform well, but its effectiveness depends on the source language, (b) low confidence factor (confidence factor means frequency of a rule divided by a number of source words where the source string of the rule occurs) yields better result than high confidence factor in particular for French and Spanish translation, and (c) the high confidence factor Transformation Rule based Translation (TRT) with *di*gram perform better than TRT with *tri*gram. The effectiveness of TRT was studied by recall and precession. Translation recall is increased as confidence factor and frequency are decreased. For French, German and Spanish recall is between 77.8% and 97.81% in the domain of bio terms and technology.

Authority Control

Matters[30] (1990) in the article "Authority Work for Transitional Catalogues" discusses a few aspects of authority work for names of person and corporate bodies and how they relate to the objectives of authority work for transitional archival catalogues. According to Mathew, the authority work in archives focus on "documentary remains" of real people, not the literary output of "bibliographic identities". The procedures of authority work in archives are as follows: (a) accepting and using heading that already established in standard authority file like LCNAF, (b) establishing new headings according to relevant standards like AACR2, (c) adding new heading to standard authority files, (d) maintaining authority data, and e) seeing that the references specified in authority records are used to facilitate access in public catalogue. The cost of authority work of person's who are not authors might increase. Headings for these names are not likely to have been established in LCNAF. Cooperative efforts will help to reduce the cost. The

distribution of personal names in archives is not studies well as like authors name in general catalogue. By that, it would be difficult to know the best policies for archival authority work. Mathew suggests that some of the authority work now done by cataloguers might be shifted to appropriately design online catalogue systems and their users.

Pappas[31] (1996) conducted a study on the authority controlled heading in a random sample of catalogue records from eight Research Libraries Information Network (RLIN) member libraries to determine the extent to which they either matched the form as established in RLIN online authority file, or in cases where no headings existed in the files, had been formulated according to AACR-2 principles and LC guidelines. The objectives of the study were: (a) identification of libraries consistently using RLIN Name Authority Record and Subject Authority Record files in their records, (b) determination of which types of errors occurred most frequently in instances where access points did not match the LC Name Authority file and Subject Authority File, and (c) using the result of this analysis to add to a list of preferred libraries kept by the Monograph Cataloging section of New York Public Library, thereby increasing productivity by modifying its copy cataloguing work flow. The findings of the study shows that: (a) although average number of errors per record were relatively small (1.64), the fact that just under half of all records (49%) contained errors in their authority-controlled fields was a discouraging findings, (b) using the average that ignored the treatment of monographic series in ascertaining percentage, only three of the eight libraries had correct-entry percentage of over 90%, (c) of all MARC fields analysed, the monographic series field accounted for the highest number of errors. The findings of the study have a number of potential practical applications. It helps to evaluating the quality of a particular record, library or network. Secondly, the study helps library for compilation of list of institutions doing quality authority work and adopting of sub routing like NYPL's NMN copy cataloguing work flows to help lower costs and process higher quantity of materials.

Calhoun[32] (1998) in the Article "A Birds Eye View of Authority Control in Cataloguing" reviews the development of community wide authority control in cataloguing, cooperative cataloguing and the working of cooperative cataloguing both from system perspective and from the perspective of a cataloger. System perspective is the process of input and output. The master copy of the authority file is housed on Library of

Congress computing system. Synchronized copies of the file are held and maintained at OCLC and Research Library Group (RLG). Catalogers at participating libraries sit at their workstations in their libraries, online to their library's catalogue or to one of the shared cataloguing system. In the course of cataloguing, cataloguers discover the need to create new authority records, or change existing ones, or capture copies of the records for use in their library catalogue. To contribute a record, the cataloguer connects to either OCLC or RLG system, adds new record or modifies the existing one. The OCLC and RLG collect the contribution and sent them to Library of Congress system once in a day. Library of Congress updates the authority file. Calhoun suggests that a new strategy and system model is perhaps needed for global library community where cultural, linguistic and practical distinction existing among national authority files. Another refinement of the current system features the development of various desktop applications to accelerate and ease the process of creating authority records. The present shared authority control system is really a service for cataloguers. There has been relatively little community wide progress on deeply integrating authority data into end-user information retrieval systems. The present authority control system covers only half of the name headings. By using automatic generation of records, we can expand authority file more economically.

Tillet[33] (2001) in the article "Authority Control on the Web" discusses the importance of authority control in the web. According to Tillet, when we add library catalogues to the mix of online resources on the web, we introduce controlled vocabularies for subjects, name and titles. It will help the users to filter their search results in the web. Authority control enables precision and recall that are lacking from today's web search. Many national and international agencies are begun to start working of international authority file. But the existence of different MARC formats and different cataloguing codes and transliteration schemes are the obstacles to attain an international authority file. The combination of Unicode and new technologies help to present all script for all languages in bibliographic and authority records. According to Tillet, authority control is gives way to access control by providing user selected display form i.e., user may prefer to see headings in their own script or user may want to see the names of works in their own language. The proposal for the creation of International Standard Authority Data Number (ISADN) not fulfilled. International authority file will facilitate sharing of authority information, simply the creation and maintenance of authority records and enable user to

access bibliographic information through controlled access. In order to develop International Authority file, Tillet propose a plan. IFLA can maintain a list of those agencies that would be willing to share their authority record and convert the authority records into a common format like Z39.50. The existing authority control number can be used to link all similar existing authority files rather than creating a worldwide system for assigning ISADN. The nineteen mandatory data elements recommended by IFLA such as LC control numbers can be used in shared authority records. Tillet concludes that authority control work remain the most expensive part of cataloguing, but through cooperative efforts like NACO, SACO and IFLA initiatives, the research done in library can be shared internationally to lower the cost

Ling Hway Jang[34] (2002) compares the difference between approaches to defining database quality in cataloguing and in online database. In cataloguing process, authority control is widely used to ensure 100% recall and precision by collocating all manifestation under one form chosen as the authoritative heading though superimposing separate elements. But many studies in online database searching shows that precision is often less of a concern among users. This is probably due to the reason that the indexing and abstracting industry has made a very minimum amount of authority control in favor of speedy database construction and flexible user interfaces. In a few databases, instead of attempting consistency among keywords through the use of subject authority control, the database system allows to use operators such as stemming or truncation in searching, or system automatically expand the plural form. However, the attempt in name authority control is rare. The 'Dialog' online database presents tagged author field as an integrated part of indexing system, which helps to partial collocation by relying on computer to gather different spellings of same name together. The indexing and abstracting industry takes a system level approach preferring a better designed and more flexible user interface and steers away from performing authority work. According to Jang, the professionals in online database construction clearly favor speedy, easy, friendly user interfaces, not authoritative, standardised, consistent entries with accuracy. The library catalogues are becoming an integral part of the increasingly larger networked information systems and system interface design has become more powerful and flexible. Therefore, it is time for cataloguers to re-examine the cost and limited nature of authority control in cataloguing

and to work with system designers to provide easier and friendlier interfaces. This will enable cataloguer to make access points less costly and search by end user will be easier.

Gentile – Tedeschi, Massino and Federica Riva[35] (2003) analyses the problems in the field of music cataloguing. Music related materials could be grouped into seven categories. (1) book on music, (2) music periodicals, (3) librettos and concert programs, (4) local history and biographies research, (5) printed music, and (6) sound recordings. According to Massino, the major problem of authority control in music occurs in music and music content related access point. The importance of controlled access by names in music is increased by the fact that music titles are often not distinctive. Names to be retrieved are not only those bearing responsibility on publication like authors but also those related to the performance (singers, players, dancers, conductors, ensembles, impresarios) and even dedicatees. Name of persons who are active in the international level may have different accepted forms in different national authority file and pose issue of transcription and transliteration. Massino points out some problems related to names. They are: (a) one man, more names, (b) one name different alphabets, (c) one man, which name? (d) two man with same name, (e) noble titles, and (f)one man, more activity. The issue of authority control on music is very much related to the elements of uniform tiles, filing tiles, medium of performance, musical form, opus number and arrangement of statement. Massiomo analyses the activities of authority control in music at international level. Authority control issues were first posed in music in the 19th century. In 1951, the International Associations of Music Library and Music Documentation Center (IAML) were founded. IAML promoted the publication of authority control activities in music. IFLA approached IAML to maintain the list of UNIMARC codes for the field of music. The author suggests that the local level research is as essential as the international co-ordination of activities.

Unlike documents for which various kinds of unique ID system exits (e.g. DOI, ISBN), the author and publication venue do not have IDs. Therefore, in current Digital Libraries like DBLP and CiteSeer, tracking bibliographic entities is not trivial. For instance, suppose a scholar changes his last name from A to B. Then, a user searching for his publication under the new name B, cannot get the old publication, although they are by the same persons. For such a scenario, it would be desirable in a Digital Libraries to track

their identities accordingly. In the paper "System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach" Hong, Yojin ;Byung –Won On and Dongwon Lee[36] (2004) presents a system level approach to solve these problems. According to Hong, the changes in author and publication entities can be grouped into three i.e., (a) Linear change: a bibliographic entity A is changed as B, (b) Spilt: a bibliographic entity is split into multiple one, and (c) Merge: two bibliographic entities are merged into one. Once name variants are identified, the findings must be inserted into Digital Library to solve the name authority control. Similarly, once duplicates are identified and put into the system, that knowledge can be exploited in searching. When such related information is updated, the system can do return merged list of display a link to publication lists under related name variants. The proposed solution is implemented in a test-bed, called OpenDBLP accessible at http://opendblp.psu.edu.

Authority Control and Cost

Authority control is an important cataloguing function aimed at achieving catalogue consistency. It is a very time consuming and labour-intensive process. During 1990s many North American academic libraries, under budgetary constrains, have trained to outsource authority control activities. Lam,Vinth-The[37] (2001) analyses the experience gained by University of Saskatchewan Library (USL) in authority control outsourcing. USL selected Library Technology Inc. (LTI) as vendor for the authority control project having the objectives: (a) cleaning up the whole database, (b) updating forms of headings, (c) protecting heading and access points, and (d) preparing the database for future annual. USL followed three steps in database clean up process: (a) extracting USL database using an 'Output Table Program' and sending it to LTI in a number of files, (b) processing of authority control files of USL database by LTI ,and (c) getting the processed files back from LTI and loading it using a 'Loader Program'. LTI took 8 weeks to complete authority control processing and loaded back the records. LTI fulfilled its guarantee of 95% success rate in database clean up.

The most expensive activity that a cataloguer can perform is authority work i.e. the establishment of one and only one form of name, series, or subject with appropriate cross-reference that guide users to the materials they seek. There are a variety of approaches to

manage cost of authority work. These include out sourcing, keeping the function in-house and some combination of these two. It is argued that name authority work for artist in in-house is more efficient. According to Boese[38] (2003), the average cost of authority work for a heading is $7.76 and for a bibliographic record is $13.19. However, depending National Name Authority Files or other important reference sources, can minimize the cost. Boese concludes that by consulting the reference source the cataloguer can create their own artist name authority records and can produce high quality records with lowest cost.

Authority Control Automation

Authority control is vital component of an online catalogue. Manual authority control procedure was replaced by creation of authority record in the online authority file. The automatic updating of authority file in online database will reduce the cost of authority work. In the article "Global Change Capabilities to improve Authority Control in an Online Catalogue" Fox and Kanafani[39] (1989) reports their experience in automation of authority work at Washington University. They developed a program as a local extension to the NOTIS software. The programs were written in the PL/I programming language. For this, they created local authority file by including additional fields in MARC format. The procedure of the automatic updating includes following steps: (1) A online request for a global change (automatic updating) results in a transaction being written to a special " batch request" file. (2) Search the transaction records against the appropriate online index for matches. (3) If match is found in the index, a transaction record is written with the key of the authority record to be processed and the key of the bibliographic record found in the index entry. (4) A replacement program reread the 4xx fields of the authority record and replaces the heading or headings in the bibliographic record. Each time a bibliographic record is changed, the modified record is rewritten to the work file. (5) A report is produced for mismatch error. (6) After manual review of records and error reports is complete, the records are reloaded into the database.

In the article, "Reengineering Name Authority Control" Snyman, and Rensberg[40] (1999) describe a model for standardisation of name in bibliographic databases. They developed a system based on International Standard Author Number (ISAN). An author has only one ISAN number, which is unique. The ISAN number works as a link among

variant name of author. The researcher developed a prototype system by using Visual Basic5. The system automatically generates ISAN number (13 digit) based on the parameters like ID number (may be social security number), Surname, Nationality, Address, Language and Telephone number. Using this ISAN, an access control file and access record file is created. The access control file consists of access records with author number (ISAN) being the authority form. Variant forms of author's name are included in the access record and linked with ISAN number. As against MARC authority record, 1xx field is replaced with numeric link (ISAN) to the access record. No distinction exists between main and added entries. Just as the authority file is currently linked with the bibliographic file, the access control file will also be linked with bibliographic file via ISAN. The system works with the help of an interface program. The user can either enter ISAN, author name or other parameter to get link to access control records. With the help of access control records, the system displays all records (publication) related to the author. If it is new publication/ author, the system automatically generates ISAN and recorded in 1xx field of access control records. By clicking on the ISAN in the interface application, the relevant access records can be displayed and name of author as it appear in the publication can be entered in the 4xx field. According to Snyman, the new system has wide implications. A national agency should monitor the creation of ISAN. Authors and publishers will have responsibility of obtaining the ISAN number. The existing MARC format will have to be adopted to accommodate new ISAN number.

Warner[41] (2001) in the paper "Automated Name Authority Control" describes a system for the automated assignment of authorised names in Levy Sheet Music Collection. Presently, this collection does not provide the traditional library "authoritative" version of a person's name and providing the ability to search on "variants" or "cross references". Thus, users searching the collection are not able to retrieve easily the varying forms of name of persons. In the proposed automated system, the Library of Congress Name Authority File can be used as tool of interoperability with Levy Sheet Music Collection or other similar collections rather than as a source of authority data. In order to test the feasibility of the project, they restricted the names of composers, lyricist and arrangers. They extracted the names from the collection using simple pattern matching techniques and grouped it into four groups i.e., seed group, two training groups, and a held-out test group. In order to retrieve authority records efficiently, they loaded the personal name

subset of the LCNAF into MySQL database. The accuracy was tested using Bayesian approach. The initial result was promising. According to Warner, the new automated system can be applied to any large collection of digital documents.

Access Control

Barnard, Linda[42] (1996) in article "A ccess Control Records: Prospects and Challenges" discusses in very practical terms about access control, and its structure and uses in future online systems. Access control is the next generation of authority records. Access control records would be linked simultaneously to bibliographic records to collocate all manifestation of a work and to other related access control records to collocate related works. In access control, instead of declaring any one as the authorized form, links the variant forms of a name. A central concept is that a library or user should be allowed to choose their preferred form of name These concepts are entirely contradictory to the second part of AACR-2. There are many other elements, which show bibliographic relationships like language in which the author has published, countries in which author has published and subject of the publication, which can be incorporated into access control records. According to Barnhart, access control records would become the intellectual locus of the online catalogue and the basis for a work- based online catalogue. The library would allow for enriched information about individuals and works, which would result in flexible but consistent indexing and retrieval.

In the article "Authority Control Simply Does Not Work" Ayres[43] (2001) demonstrates through case studies how authority control does not work. According to him, the performance of authority control is in between two hypotheses, i.e., authority control works better in theory than in practice and authority control simply does not work. With the cheep computing power available toady, the catalogue users should expect every known variation of a catalogue access point to have a cross reference to the heading. The sad fact is that cross-references used in library catalogues are highly selective and not comprehensively. Ayres study implies that the power and potential given to the library catalogue by authority control does not work unless cross-references are used comprehensively. Ayres conducted 8 case studies (key word searches) in LC OPAC. They are: (1) *Dostoyevsky*: the name 'Dostoyevsky' retrieved 80 downloads. An another search

under its variants retrieved 'Dosoyevsky' (329), 'Dostoevskii'(223), 'Dostoevski'(5) results. If the authority control mechanism works well, all other variations would be linked to the heading. (2) *Creativity*: a search in title contains 'creativity' resulted 1259 records. Using the 'select subject' options on retrieval of 400 of these records showed that 398 headings were used. Of these only 2 headings and 4 records used 'creativity' while 10 headings and 158 records have the word 'creative ability'. (3) *Smoking*: the word 'smoking' searched both in 'subject contain' option and 'title contain' option produced 1145 and 891 results respectively. From 80 records in 'title contain' eleven records had no subject containing 'smoking'. There were however subject entries under cigarette habit, cigarette smoking. The other case studies are (4) Commuings E (full name), (5) sexual harassment (different aspects), (6) Secularizaiton (alternative spellings), (7) NFPA (abbreviations), and (8) Health insurance (inverted heading) produced similar results. Ayres concludes that giving comprehensive cross-references is better solutions to solve these problems.

Conclusion

The researcher reviewed 43 articles relating to the various aspects of authority control. It presents the issues relating to authority control and its importance in information retrieval. These studies are very helpful to formulate new policies and solutions to meet the requirements modern Information Retrieval Systems.

REFERENCES

1. Beheshti, Jamshid. "The evolving OPAC" *Cataloging and Classification Quarterly* 24.1/2 (1997):163-185.

2. Sridhar M.S. "OPAC Vs Card Catalogue: A Comparative Study of User Behavior" *The Electronic Library* 22(2004): 175-183.

3. Krieger, Mitchael T. "Characteristics of the 670 Field in Records for Names in the Anglo-American Authority File." *Cataloging and Classification Quarterly* 23.1 (1996): 99-119.

4. Johnson, Bruce Chr. "XML and MARC: Which is "Right"?" *Cataloging and Classification Quarterly* 32.1 (2001): 81-90.

5. Fiander, David J. "Applying XML to the Bibliographic Description." *Cataloging and Classification Quarterly* 33.2 (2001):17-28.

6. Mansor, Yushiana. "Bibliographic Exchange in Malaysia: Variations in Name Headings" *Library Review* 52.1(2003): 38-42.

7. Banerjee, Kyle. "Describing Remote Electronics Documents in the Online Catalogue: Current Issues" *Cataloging and Classification Quarterly* 25.1(1997): 5-20.

8. Ruiz-Perez, R. "Consequence of Applying Cataloging Codes for Author Entries to the Spanish Library Online Catalogue" *Cataloguing and Classification Quarterly* 32.3 (2001): 31-35.

9. Vassie, Roderic. "Improving Access in Bilingual, Biscript Catalogues Through Arabised Authority Control." *Online Information Review* 24(2000): 420-428.

10. Abduoulay, Kaba. "Perception of Cataloguers and End User Towards Bilingual Authority File." *The Electronic Library* 20 (2002): 202-210.

11. Olson, Chalermsee. "Cataloging Southeast Asian Language Materials: the Case of the Thai Languages" *Cataloging and Classification Quarterly* 22.2 (1996): 19 – 28.

12. Yewang, Wang, "A look into Chinese Persons' Names in Bibliography Practice" *Cataloging and Classification Quarterly* 31.1 (2000): 51-81

13. Khurshid, Zahiruddin. "Arabic Script Materials: Cataloging Issues and Problems" *Cataloging and Classification Quarterly* 34.4(2002) 67-77.

14. Plettner, Martha Sceirs 2003. "Arabic Name Authority in the Online Environment: Options and Implication" 5 March. 2005 <http://www.uni-amberg.de/unibib/melcom/PlettnerICBC.html>

15. Arbabi, M. *et al* "Algorithms for Arabic Name Transliteration." *IBM journal of Research and Development* 38 (1994): 183-205

16. Stalls, Bonnie Glover, and knight, Kevin.1998."Translating Names and Technical Terms in Arabic Text"10March2005 <http://acl.ldc.upenn.edu/W/W98/W98-1005.pdf>

17. Jeong, Kil Soon *et al.* "Automatic Identification and Back-transliteration of Foreign Words of Information Retrieval." *Information Processing and Management.* 35 (1999): 523-540.

18. Nasreen, Abdul Jaleel and. Leah S Larkey 2002. " English to Arabic Transliteration for Information Retrieval : A Statistical Approach "*CIIR Technical Report* 10March 2005 <http://ciir.cs.umass.edu/pubfiles/ir-261.pdf>

19. Nasreen, Abdul Jaleel and Leah S. Larkey 2003. "Statistical Transliteration for English- Arabic Cross Language Information Retrieval" *Proceedings of the Twelfth International Conference on Information and Knowledge Management* 10March 2005 <http://ciir.cs.umass.edu/pubfiles/ir-293.pdf>

20. Pfeifer, Olrich, ,Thomas Poersch and Norbert. Fuhr "Retrieval Effectiveness of Proper Name Search Methods" *Information Processing and Management* 32(1996): 667-679.

21. Lewellen, Mark. 1999."Name Searching with Artificial Neural Network" *Workshop on Natural Language Processing and Neural Networks* 10March 2005 <http://www.math.ryukoku.ac.jp/~qma/activity/NLPNN99/>

22 Larkey,Leah S, Nasreen Abdul Jaleel and,Margaret Connell. 2003."What is in a Name?: Proper Names in Arabic Cross Language Information Retrieval " *CIIR Technical Report IR 278* 10 March 2005 <http://ciir.cs.umass.edu/publications/>

23. Spink,Amanda and Bernard J. Jansen. "Searching for People on the Web Search Engines" *Journal of Documentation* 60(2004):266-278

24. Weintraub, Tamara S. "Personal Name Variations: Implications for Authority Control in Computerized Catalogues" *Library Resources and Technical Services* 35(1991): 217-228.

25. Strunk, Kirsten."Control of Personal Names "*Cataloging and Classification Quarterly.* 14.2(1991):63-79.

26. Taylor, Arlene G. "Variations in Personal Name Access Points in OCLC Biographic Records" *Library Resources and Technical Services* 36(1992): 224-241.

27. Kock, Gideon de V. de "Searching on Full Name Providing for Spelling Variations" in *Proceedings of the 2002 Annual Research Conference of the South African Institute for Computer Scientists and Information Technologists on Enablement Through Technology*, 2002 .255-255.

28. Raghavan, Hema and James Allan 2004. " Proper Names and Their Spelling Variations in Automated Speech Recognition Output" 10 March 2005 http://ciir.cs.umass.edu/pubfiles/ir-361.pdf.

29. Toivonen,Jarmo *et al* "Translating Cross-Lingual Spellings Variations Using Transformation Rules" *Information Processing and Management* 41(2005):859-872.

30. Matters, Marion." Authority Work for Transitional Catalogues."*Cataloging and Classification Quarterly.* 11(1990): 53-69.

31. Pappas, Evan. "An Analysis of Eight RLIN Member's Authority Controlled Access Points for Purpose of Speeding Copy Catalogue Workflow" *Cataloging and Classification Quarterly* 22.1(1996):29- 47.

32. Calhoun, Karen.1998."A Birds Eye View of Authority Control in Cataloguing", *Proceedings of the Taxonomic Authority File Workshop* 10 March 2005 <http://www.calacademy.org/research/informatics/taf/proceedings/Calhoun.html >

33. Tillet, Barbara. 2001."Authority Control on the Web" *Proceedings of the Bicentennial Conference on Bibliographic Control for the New Millennium* 10 March2005<http://www.loc.gov/catdir/bibcontrol/tillett_paper.html

34. Ling Hwey Jang. "What Authority? What Control? *Cataloguing and Classification Quarterly* 34.4 (2002): 91-97.

35. Gentile–Tedeschi, Massino and Federica. Riva, 2003. " Authority Control in the Field of Music: Names and Titles" *Proceedings of International Conference on Authority Control* 10March 2005 http://www.unifi.it/universita/biblioteche/ac/ relazioni/gentili-tedeschi_eng.pdf>

36. Hong, Yoojin, Byung-Won On and Dongwon Lee. "System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach" in *Lecture Notes in Computer Science*: Springer-Verlag, 3232(2004): 134-144.

37. Lam, Vinth-The. ."Outsourcing Authority Control: Experience of the University of Saskatchewan Libraries." *Cataloging and Classification Quarterly*.32.4 (2001) 53-69.

38. Boese, Kent C. "What in a Name: Associated Cost of Authority Work for Artist Names: the Bottom Line" *Managing Library Finance* 16.3 (2003): 106-110.

39. Fox, Judith A and Kay Kanafani, "Global Chang Capabilities to Improve Authority Control in an Online Catalogue." *Information Technology and Library* 8(1989): 273-283.

40. Snyman,MMM and Rensberg, Jansen Van. "Reengineering Name Authority Control" *The Electronic Library* 17 (1999):.313-322.

41. Warner, James W.and Elizabeth W. Brown. "Automated Name Authority Control" in *Proceedings of the First ACM/IEE-CS Joint Conference on Digital Libraries*: USA (2001) :21-22

42. Barnhart, Linda. 1996. "Access Control Records: Precepts and Challenges " *Authority Control in 21st Century: An invitational Conference* 10March 2005 <http://digitalarchive.oclc.org/>

43. Ayres, FH. "Authority Control Simply Does Not Work" *Cataloging and Classification Quarterly* .32.2(2001): 49-59.

Chapter 4

METHODOLOGY

A unique name in Arabic may have variant spellings in English occurred by transliteration process. The proposed study is intended to understand the spelling variation in usage of Arabic name and the influence of regional languages in transliteration process. This study can be done in two methods. One is listing all unique Arabic names and its spelling variations i.e., name approach. The second method is to compare Arabic alphabet with corresponding/similar Roman letters and find out the variations with the help of standardized Romanization Table i.e., spelling approach. The researcher selected the second method. The proposed study is intended to compare standardized Romanization table and usages by the people. The plan of the study can be diagrammatically represented as below:

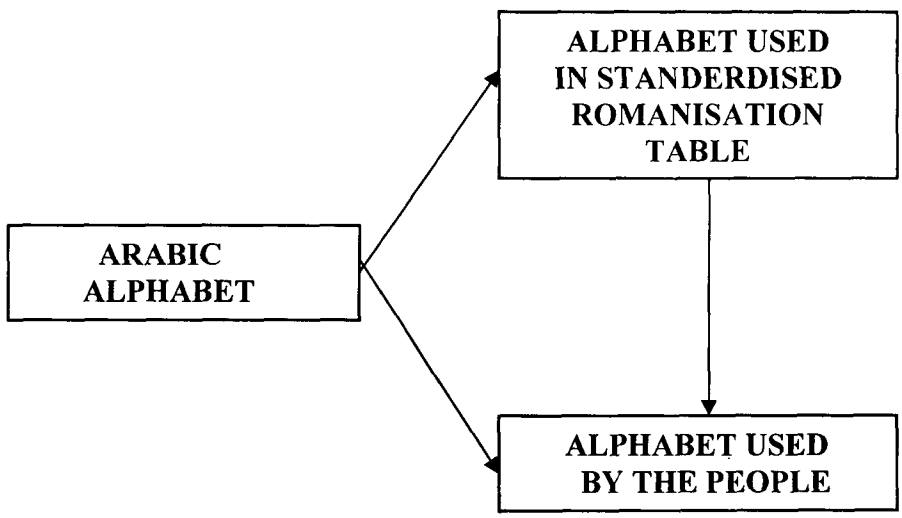


Figure 4.1: Diagram of the Plan of Analysis

The alphabets are grouped as consonants and vowels. The Library of Congress Romanisation Table (LCRT) has taken standardized Romanisation table. All the personal names (name of authors) selected as example were lived/living personalities, which are used as headings in Library of Congress Name Authority File (LCNAF). The LCRT is attached in Appendix I.

There are 28 alphabets in Arabic and 26 alphabets in English. Among these, B(ب), H (ح), D(د), R(ر), K(ك), L(ل), M(م), N(ن) are the correct pronunciation/ transliterations of corresponding Arabic alphabets. The other alphabets are slight difference in pronunciations. Even though the alphabets like F(ف), S(س), SH(ش), Z(ز),W(و) are the correct pronunciation of corresponding Arabic alphabets, they are also used to represent the other Arabic letters.

Generally, authority file is created for individual authors. It means that an authority file created for one author may not be same as another author having the same name. The proposed study is not based on any individual authors who are using different names/spellings, but it is based on unique Arabic names and its variant spellings used by the people. Only 'ism' (personal names) is included in the study. The other elements like 'kunya''laqab' are excluded from the purview of the study.

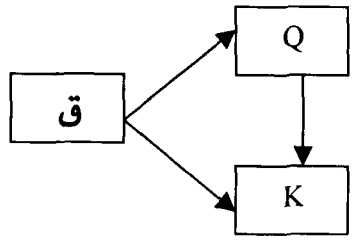
Chapter 5

ANALYSIS

In this chapter, the Arabic alphabets along with corresponding standardized Roman alphabet were compared with alphabet used by the people. The alphabets are grouped as consonants and vowels.

Consonants

K and Q



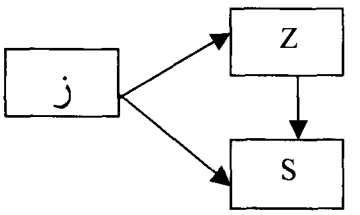
The letter K and Q are simultaneously used in transliterating Arabic letter ق. As per the Library of Congress Romanization Table (LCRT) ق is transliterated as Q The ق may come on the beginning or at the end of the name like قادر (Kader),قاسم (Qasim) إسحاق (Ishaq).Some examples are given in the Table 5.1.

Table 5.1: Names Spelt by K or Q

Name	Name in Arabic	Name as Headings in LCNAF
Qadir	قادر	Hamza, Abdel **K**ader. **Q**adir, Abdul, Sir, 1874-1951
Ishaq	إسحاق	Ishaq, Ashfaq, 1953 Ishak Efendi, 1679-1734
Qasim	قاسم	**Q**asim, Abd al-`Az¯iz, 1933 **K**asim, Anwar
Tariq	طارق	Tariq, Abdur-Rahman, 1915- Tarik, Hid¯ayat `Al¯i Najaf¯i, 1894-939
Farooq	فاروق	Farook, Hossain Muhammad Farooq, Abdul Aziz

The regional languages are highly influencing in transliteration of Arabic letter ق For example, in Malayalam ق is transliterated as ക/ഖ like Kader as കാദർ/ഖാദർ and Qasim as കാസീം/ഖാസീം. Generally ഖ is used for ق at the end of the Malayali muslim names as in رفيق / Rafeeq/ റഫീഖ് and إسحاق / Ishaq / ഇസ്ഹാഖ്. So while writing the above mentioned Arabic names from Malayalam into English, they use K for ക and Q for ഖ. The spelling variation is occurred due to the influence of regional languages.

S and Z



The Arabic letter س and ز are Romanised as S and Z respectively. The pronunciation of س and S, ز and Z are more or less similar. This two letters are not confusing to transliterator. However, transliteration of these two letters through the regional languages creates variant spellings. Some names are given in Table 5.2.

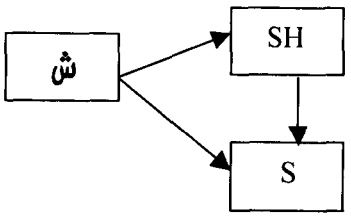
Table 5.2: Names Spelt by Z or S

Name	Name in Arabic	Name as Headings in LCNAF
Zuhra	زهره	Suha_ra, Bi. Em Nur Nina Zuhra
Zaid	زيد	Zaid, Abdulla S., 1945- Said, A. Fuad, 1924-
Zuhair	زهير	Suhair Cunkatta_ra Zuhair, Vah¯id
Zubair	زبير	Zubair, Shah Muhammad, 1884- Subair Vempall¯ur, 1959-
Aziz	عزيز	Asis Aziz, Abdul, 1884-.

In Malayalam,both س and ز is represented by സ only. There is no equivalent to Z / ز in Malayalam. So, when transliterating Malayali Muslim names having the letter ز, സ is used which may be again transliterated as S in English instead of Z. Here are some examples.

Arabic	Malayalam	English
عزيز	അസീസ്	Asis instead of Aziz
زين لدين	സൈനുദ്ധീൻ	Sainuddin instead of Zainuddin
زيد	സൈദ്	Said instead of Zaid.

S and SH



The Arabic letter ش is Romanised as SH and س is Romanised as S. For example, بشير as Bashir and سليم as Saleem. However, it is found that some people are using S to represent ش in Arabic. Here are some examples in Table 5.3.

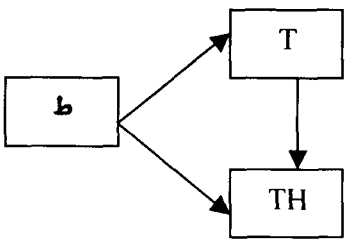
Table 5.3: Names Spelt by S or SH

Name	Name in Arabic	Name as Headings in LCNAF
Shahida	شاعده	**S**¯ahid¯a Begama, 1955- **Sh**¯ahidah Begam.
Ashraf	أشرف	**Ash**raf, A. B., 1935- Asraf Hj. A. Wahab, 1927-
Shamsuddin	شمس لدين	**Sh**amsuddin, AbulKalam M. Samsuddin, M.
Shihab	شهاب	**Shih**¯ab, `Abd al-Q¯adir Muhammad Sih¯abudd¯in Poyttumkatav, 1963-
Shamir	شمير	**Sh**amir Hassan, S. Sameer, Ahmed Firoze

The variation in spelling occurs due to the influence of regional languages. In Malayalam, ش is represented by the letter ഷ /ശ. For example, Bashir as ബഷീർ/ബശീർ. When people writing their name in English they use S or SH as they like Here are some names in Malayalam.

Arabic	Malayalam	English
اشرف	അഷ്റഫ്/അശ്റഫ്	Ashraf/Asraf
شمير	ശമീർ/ഷമീർ	Samir/Shamir

T and TH



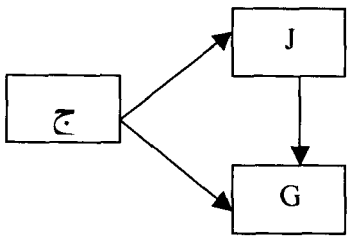
The narrow differences in pronunciation of Arabic letters ط and ث are the reason for spelling variation of T and TH. As per the LC Romanization Table, ت is Romanised as T , ط as T̲ and ث as TH. However people uses TH to represent ط in Arabic. Here are some examples in Table 5.4.

Table 5.4: Names Spelt by T or TH

Name	Name in Arabic	Name as Headings in LCNAF
Mustafa	مصطفي	Musthafa Kamal Pasha, 1939- Nas¯ir, Mustafa Abd al-Maj¯id
Latief	لطيف	Latief, Abdul, 1926- Lathief, A. K.
Talib	طالب	T¯alib, `Abd al-Rahm¯an bin Ahmad Thalib, Mosthamir, 1963
Talha	طلحة	Talhah Abd. Latiff Thalha, Mohmed
Fathima	فاطمه	Kapil, Fathima Kutty Fatima,C

In Malayalam, the Arabic letter ت and ط are represented by the letter ത and ത്വ respectively. For example, Musthafa as മുസ്തഫ Lathief as ലത്വീഫ് and Taha as ത്വാഹ. There is no equavalant to ث / TH in Malayalam. The correct pronunciation of ط in Malayalam is ത്വ . However, it is used only in few names. So, when transliterating ط and ت into English through regional languages, peoples are using T and TH simultaneously and it would cause spelling variations.

G and J

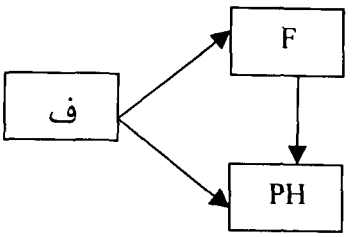


G is rarely found transliterated form of Arabic letter ج. The pronunciation of ج is very much similar to J in English. The practice of using G for ج is found in Egyptian names. For example, Gamal Abdul Nasar , Gemal-el-Dine Pacha. Some other examples are given in Table 5.5.

Table 5.5: Names Spelt by J or G

Name	Name in Arabic	Name as Headings in LCNAF
Jabir	جابر	J¯abir ibn Hayy¯an. G¯abir, Hayy¯an ibn
Jamal	جمال	Gamal-Eldin, Saad M Jamal, `Abd al-Fatt¯ah, 1938
Jalal	جلال	Jal¯al, `Abd al-`Az¯iz `Abd All¯ah Galal, Ahmed, 1948

F and PH

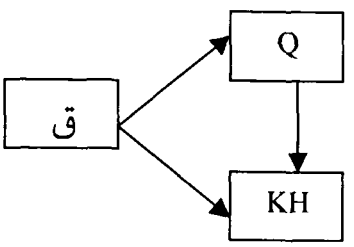


PH is the rarely found transliterated form of Arabic alphabet ف, which is usually Romanised as F. The practice of using PH to represent Arabic letter ف is widely found in Egypt and Malaysia. Here are some examples in Table 5.6.

Table 5.6: Names Spelt by F or PH

Name	Name in Arabic	Name as Headings in LCNAF
Mustafa	مصطفي	Mustap**ha**, Abdul Raufu Musthafa.
Latif	لطيف	Latt¯i**ph**, N. K. A., 1936 Lathiful Khuluq, 1968-

Q and KH



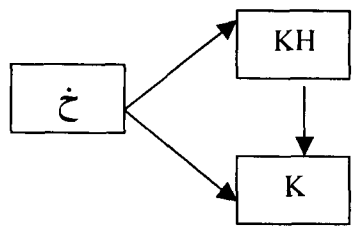
As per the LC Romanization table the Arabic letter ف is transliterated as Q and خ is transliterated as KH like قاسم as Qasim and خالد as Khalid. But in practice people uses KH instead of Q to represent the Arabic letter ق (Q). Some examples are given in Table 5.7

Table 5.7: Names Spelt by Q or KH

Name	Name in Arabic	Name as Headings in LCNAF
Qadir	قادر	**Kh**ader, Naser, 1963- **Q**¯adir, Ahs¯an, 1912-1969
Qasim	قاسم	**Kh**¯azim, `Al¯i Hasan **Q**asim, `Abd al-Satt¯ar
Qamar	قمر	**Kh**amar, Ali. **Q**amar, 1901-1951

The spelling variations shown above occur due to the influence of regional languages in transliteration process. In Malayalam, the Arabic name قادر /Qadir is transliterated as ഖാദർ. But there is no equivalent to Q in Malayalam. So, while people transliterating their name into English from Malayalam, they uses KH to represent the Malayalam letter ഖ.

KH and K



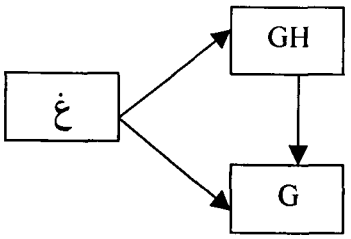
It is a common spelling variation. The Arabic letter خ is transliterated as KH. In practice, some names having the letter خ is transliterated as K. Some examples are given below.

Table 5.8: Names Spelt by KH or K

Name	Name in Arabic	Name as Headings in LCNAF
Khaleel	خليل	**K**aleel, M. C. M. **Kh**aleel, Raz (Raziuddin)
Khalid	خالد	Haun,Wadih **K**alid Atallah **Kh**alid Abd al Aziz,1966
Khadija	خديجه	**K**adija, Refik **Kh**adija Mohamed Awaleh

In Malayalam, ഖ is more accurate transliteration of خ. But, some people are using 'ക' to write their name, as they like.

GH and G

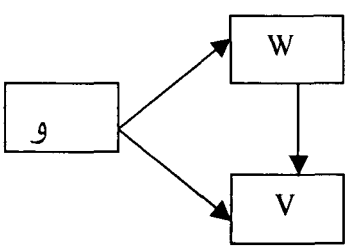


The Arabic letter غ is transliterated as GH in English. However, it is also seen that some people are using G to represent the same. For example,

Table 5.9: Names Spelt by GH or G

Name	Name in Arabic	Name as Headings in LCNAF
Ghafoor	غفور	Abdul **G**afoor, A. L. M. **Gh**afoor, Abdul
Ghalib	غالب	**G**alib, Hamid **Gh**alib, Asadullah
Ghania	غانيه	**Gh**ania, Maleem Mahmoud **G**ania, Mahmoud, 1950-
Ghani	غاني	**G**ani, A. B. M. Osman **Gh**ani, Abdul, 1941

V and W



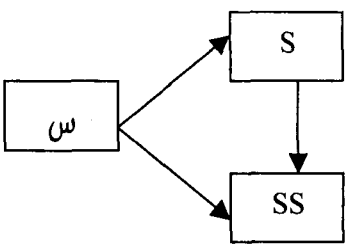
The Arabic letter و is transliterated as W in English. However, it is also seen that people are using V to represent و. Some examples are given in Table 5.10.

Table 5.10: Names Spelt by W or V

Name	Name in Arabic	Name as Headings in LCNAF
Anwar	أنور	Anvar `Al¯i, Sayyid, 1928- Anwar, Abdul Aziz
Munavvar	منوّر	Munavvar Lakhnav¯i, 1897-1970 Munawar, Mohiuddin Munawwar, Muhammad, 1923
Vahida	واحد ه	Nainar, Vahida Wah¯idah, Subh¯i
Javad	جواد	Jawad, Abdallah, 1946- Javad, Ahmad, 1892-1937
Tanvir	تنوير	Tanw¯ir, Muhammad Hal¯im Tanvir, 1968-

The above mentioned problem arises due to the similarity of pronunciation in between V and W.

S and SS



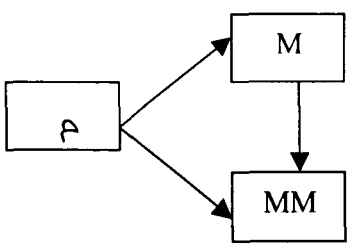
SS is the emphasized form of S. The Arabic alphabet س is Romanised as S. However, in some names, it is found that double S is used to represent the Arabic letter س, when that letter is stressed in pronunciation. In Arabic-English transliteration process, 'double letter' is used to represent emphasized form of Arabic letter. For example س as S سّ as SS, م as M and مّ as MM. But in practice double letter are used in the place of single letter. A few examples are given in Table 5.11.

Table 5.11: Names Spelt by S or SS

Name	Name in Arabic	Name as Headings in LCNAF
Husain	حسين	Husain, A. B. M., 1934 Hussain, A. Imtiaz, 1953
Masoud	مسو د	Massoud, Ahmad Shah Masouduzzafar.
Yousef	يوسف	Yousef, Ahmed, 1950- Youssef, Abdel Fattah
Jasir	جاسر	Al-Jasser, Jasser Abdulrahman. J¯asir, Muhammad T¯ah¯a, 1928-
Qasim	قاسم	Abdullah, Qassim Abdul-Jaleel. Q¯asim, `Abd al-Kar¯im, 1914-1963

In Malayalam the letter S is transliterated as സ and almost all names having the letter is using S. For example, حسين /Husain/ഹുസൈൻ يوسف/Yousef/ യൂസഫ്. But, the name Hasan is written as ഹസ്സൻ by the Malayalies.

M and MM



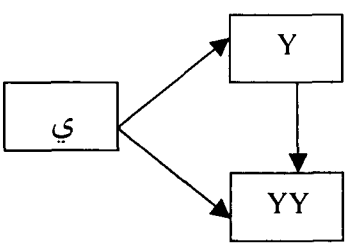
The Arabic letter م is transliterated as M and مّ, the emphasized form of م, is transliterated as MM. However, there is practice of using M to represent مّ (MM) and MM to represent م(M). Some examples are given in Table 5.12.

Table 5.12: Names Spelt by MM or M

Name	Name in Arabic	Name as Headings in LCNAF
Muhammad	محمّد	Muha**mm**ad, A. Rasyid, 1953 Muha**m**ad, Ali
Muzammil	مزمّل	Muza**m**il, A. R. Muza**mm**il, Mohd., 1953-
Ahmad	أحمد	Ah**m**ad, `Abb¯as, 1923-1978. Aha**mm**ad, En. I. E., 1932-1989

The name Muhammad is common name among Muslims and by that, a number of variant spellings are found in writing this name. There are 28 variant forms of 'Muhammad' in use. This may be influence of the regional languages and pronunciation differences in the transliteration process. In Malayalam, Muhammad is transliterated as മുഹമ്മദ് with മ്മ (MM). At the same time, the name Ahmad is transliterated in two ways ie അഹമ്മദ് / അഹമദ് with single and double M.. For example ഇ.അഹമ്മദ് (E. Ahammad M.P). This spelling variation has occurred due to the pronunciation difference.

Y and YY



YY is the emphasized form of Y. The Arabic letter ي is transliterated as Y and يّ as YY. However, while transliterating the name having يّ people may use Y and vice versa. Some examples are given in Table 5.13.

Table 5.13: Names Spelt by Y or YY

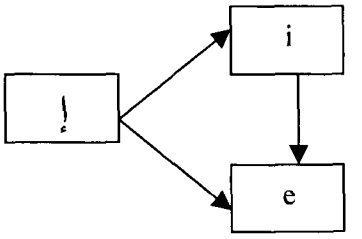
Name	Name in Arabic	Name as Headings in LCNAF
Ayub	أيّوب	Ayub, Mahmood Ali, 1948- Ayy¯ub, `Abd al-Razz¯aq
Dayyani	ديان	Dayy¯an¯i, Bihn¯am Day¯an¯i, Bihn¯am
Khayam	خيام	Omar Khayyam Khayam, Massoud
Hayyan	خيان	Hayy¯an, Muhammad Hayane, Omar
Dayyub	ديّوب	Day¯ub, Suhayr Y¯usuf Dayy¯ub, Mahm¯ud Radw¯an

In Malayalam, the name Ayyub is written as അയ്യൂബ°/അയൂബ° as similar in English.

VOWELS

Majority of spelling variations in transliteration process occurs in vowel transliteration. In Arabic, there are three vowels and in English, it is five. While transliterating Arabic name into English, people are using different combinations of English vowels. This kind of spelling variations occurs in between 'i' and ie, 'u' and 'o', 'i' and 'y' (y is not a vowel, but the pronunciation is very similar to 'i'), 'i' and 'e', 'u' and 'ou', 'e' and 'u', 'oo' and 'u'. The major reason for such kind of spelling variations is the pronunciation difference of Arabic name by non-Arabic people.

'i' and 'e'



'i' and 'e' are vowels in English and there are corresponding vowels in Arabic also. Some Arabic name starting with 'i' are transliterated as 'e' also. For example,

Ibrahim as Ebrahim. This practice is predominantly seen in African countries. Some examples are given below:

Table 5.14: Names Spelt by I or E

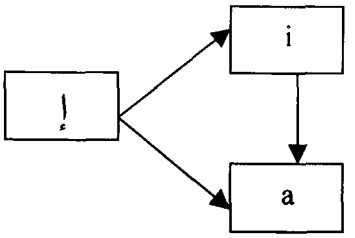
Name	Name in Arabic	Name as Headings in LCNAF
Ibrahim	ابراهيم	**E**brahim, Moosa. **Ib**r¯ah¯im, `Abd al-`Al¯im
Ismail	إسماعيل	**Ism**¯a`¯il, `Abd al-Fatt¯ah **E**smail, Aziz
Ishaq	إسحاق	**Es**`haq, Mohammad **Ish**aq, Ashfaq, 1953
Ilyas	إلياس	**El**yas Omar **Ily**¯as Ahmad, 1891-1960.
Irfan	عرفان	**Er**fan, Niaz **Hab**ib, Irfan, 1931-

This is the influence of regional languages and culture. The same case may occur in the middle of the name. But it is a common spelling variation among transliterated Arabic names. This kind of spelling variations occurs due to the vowel transliteration. A few examples are given in Table 5.15.

Table 5.15: Names Spelt by E or I

Name	Name in Arabic	Name as Headings in LCNAF
Nasir	ناصر	**N**aser, Saleh A. **N**¯asir, `Abd All¯ah
Salih	صالح	**S**¯alih, `Abd al-Q¯adir, 1908- **Sal**eh, Abdul Rachman, 1943-
Qadir	قادر	**Qad**ir, Asghar **Qad**er, Shaik A., 1932-
Hashim	عاشم	**Hash**em, Abul. **Hash**im, A. S. 1927-
Hamid	حا مد	**Ham**ed, Amir, 1962- **Ham**id, A. Shamad.

'i' and 'a'



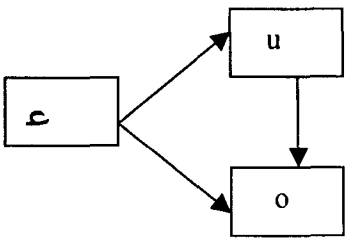
It is a common spelling variation among transliterated Muslim names. It occurs due to pronunciation difference and the influence of regional languages. Here are some examples in Table 5.16.

Table 5.16: Names Spelt by A or I

Name	Name in Arabic	Name as Headings in LCNAF
Nasir	ناصر	Nasar, S. A. Nasir, A. S., 1928-
Yasir	ياسر	Yasar, Izzet, 1951- Yasir Abdul Rahman, 1957-
Qadir	قادر	Qadir, Abdul, Sir, 1874-1951 Qadar Bilgr¯am¯i, 1833-1883.
Jasir	جاسر	Jas¯ar, `Al¯I J¯asir, Muhammad T¯ah¯a, 1928-
Tahir	طاهر	Tahar, Ahmed T¯ahir, `Abd All¯ah, 1955-

In Malayalam the name ناصر (Nasir) is written in two ways i.e., നാസർ/നാസീർ Same is the case in the other names mentioned above.

'o' and 'u'

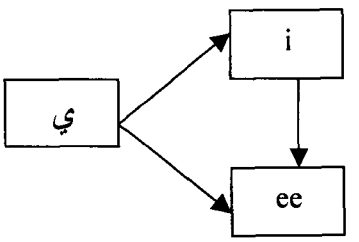


Some Arabic names starting with vowel ء (u) are widely transliterated with variant spellings. Some people use 'o' instead of 'u'. For example, عمر as Umar/Omar. A search in World Biographical Index shows total 58 hits under Othman and 27 hits under Uthman. As per the LC Romanization Table, ء is transliterated as u. But regional languages influence the transliteration process and people uses different spellings. For example, the name Uthman with spelling Othman is widely used in Malaysia. The name Umar with spelling as Omar is widely seen in countries like Egypt, Espana and Brune. Also, the name starting with أ (u) is transliterated as 'o'. A search in Google under Osama bin Ladan retrieved 92% while under Usama bin Ladan retrieved only 4%. Here are some names with spelling 'o' and 'u'.

Table 5.17: Names Spelt by O or U

Name	Name in Arabic	Name as Headings in LCNAF
Uthman	عثمان	**O**thman Alhabshi, Syed **U**thm¯an, `Abd al-Rahm¯an
Umar	عمر	**O**mar, Abdullah **U**mar, `Abd al-Rah¯im, 1929-
Ubaidullah	عبيد الله	**U**baidullah, M., 1957- **O**baidullah, A. K. M
Usama	أسامة	Bin Laden, **O**sama, 1957- **U**s¯amah ibn Munqidh, 1095-1188
Uwais	أويس	**O**wais, Naim **U**wais, Mohammed Lawal

'i' and 'ee'



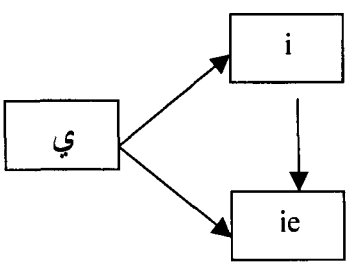
The Arabic vowel ِ is transliterated as 'i' and ي as 'ee'. For example, جاسِر as Jasir and مجيد (م ج ي د) as Majeed.. However, people use 'i' instead of 'ee' to represent ي in their names. So, they write Majeed as Majid. Some examples are given below:

Table 5.18: Names Spelt by I or EE

Name	Name in Arabic	Name as Headings in LCNAF
Habeebah	حبي به	H¯ab¯iba, ¯Ahas¯ana, 1917-1985 Habeeba Zubair
Saleem	سليم	Saleem Akhtar, 1934- Salim, A. K.
Anees	أنيس	An¯is, `Abd al-`Az¯im. Anees, Munawar A.
Hameed	حميد	Hameed, Abdul, 1908- Ham¯id, C¸¯ennamangal¯ur
Haneefa	حنيفة	Haneefa, M. Mohammed. Hanifah, Abu

This kind of spelling variation occurs due to pronunciation difference and the influence of regional languages.

'i' and 'ie'



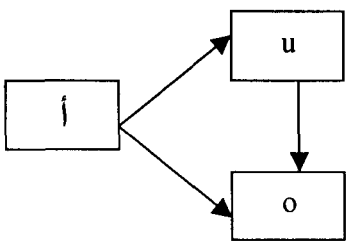
'i' and 'ie' are simultaneously used to represent ي (ya) in Arabic. This kind of spelling variation occurs in the name having the vowel ي (ya) in the middle or at the end of the name. Some other examples are given in Table 5.19.

Table 5.19: Names Spelt by I or IE

Name	Name in Arabic	Name as Headings in LCNAF
Bashir	بشير	Bashir, Abdullahi Bashier, Salman H., 1964-
Lathief	لطيف	Khuluq, Lathiful, 1968- Lathief, A. K.
Faried	فريد	Far¯id, `Abd al-Rahm¯an, 1919- Faried, Samir
Zaid	زيد	Zaied, Abdalla A. Zaid, Abdulla S., 1945
Hussien	حسين	Hussin, Jabbar Yassin, 1954- Hussien Alattas, Syed

This kind of spelling variation is occurs due to the pronunciation difference in regional languages.

'u' and 'o'



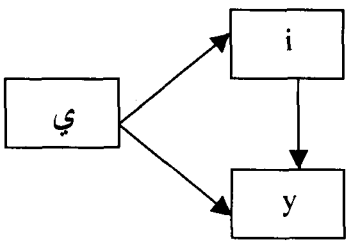
The Arabic vowel أ is transliterated as 'u' in English. Almost all transliteration table uses 'u' to represent Arabic vowel أ The name محمّد (م ح م د) (Muhammad) is transliterated as Muhammad as per the transliteration table. But in practice, a study reveals that 32% of the people having the name Muhammad use 'Mohammad' in writing. This may the influence of different culture .For example a search for Muhammad in World Biographical Database retrieved 981 records while mohammad retrieved 701 records.

Among 701 records 98% persons having to Indian subcontinent. Similarly a search for the name Hossain retrieved 30 entries. Among these 27 are/were Pakistanis 2 Indians and 1 others. Here are some names having the spelling 'u' and 'o'.

Table 5.20: Names Spelt by U or O

Name	Name in Arabic	Name as Headings in LCNAF
Muhammad	محمّد	Muhammad, `Abd al-Haf¯iz Mohammad, A. Zayed
Hussain	حسين	Hussain, Ahmed, Ustad Hossain, Anwar, 1948-
Musthafa	مصطفي	Mostafa, Javed, 1966- Mustafa`Abd al-Mun`im Fahm¯i.
Mansur	منصور	Mans¯ur, `Abd al-`Az¯im. Mansor Ahmad Saman, 1949-
Mubashir	مبشّر	Mubashir Ali, Sayd Mobashar, Mahr Ghulam Ali

'i' and 'y'



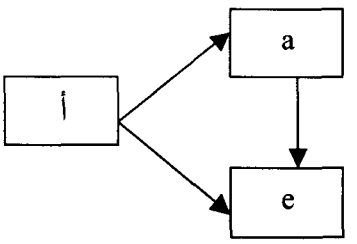
The Arabic vowel ي is transliterated as 'i' and 'y' in English. People are also using 'i' and 'y' simultaneously to represent ي in Arabic. A few examples are given in Table 5.21

Table 5.21: Names Spelt by I or Y

Name	Name in Arabic	Name as Headings in LCNAF
Yahya	يحيي	Yahia, Mohammed Haj Yahya`Abd al-Ham¯id, 1935-
Saif	صيف	Sayf, `Abd All¯ah, 1946- Saif, Ahmed A
Aisha	عايشه	Noor Ayesha. Aisha Akbar
Feisal	فيسل	Feisal, Yusuf A. Faysal, `Abd All¯ah, 1922-
Ubaid	عبيد	Ubayd, Ahmad `Abd al-Ghaff¯ar. Ubaid, Ti. Ke.

This may happen due to the similarity of pronunciation of 'i' and 'y'. In Malayalam the name Saif is transliterated as ൈസഫ്/സയ്ഫ് .The regional language also influence the transliteration process.

'a' and 'e'

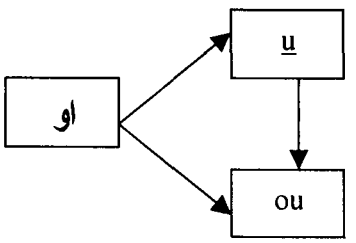


It is a common spelling variation while transliterating Arabic name having the vowel أ (a). Every letter in Arabic with vowel أ (a) is transliterated as 'ba'(ب) 'tha'(ت), 'ja' (ج) etc.,. However in practice, people are using 'e' instead of 'a' in writing their names in English. For example ر ب ك أ (أكبر) as Akbar / Akber. In Malayalam, it is written as അക്ബർ with short vowel i.e., Akber. A search in World Biographical Index retrieved 62 entries under Akbar while 6 in Akber. Other examples are given in table 5.22

Table 5.22: Names Spelt by A or E

Name	Name in Arabic	Name as Headings in LCNAF
Muhammad	محمّد	Muhammad, A. Rasyid, 1953- Muhammed, Idris N., 1952-
Ahmad	أحمد	Ahmad, `Abb¯as, 1923-1978 Ahmed, A. Karim, 1939
Akbar	أكبر	Kakkattil, Akbar, 1954- Akber, Md. Ali, 1948
Mahdi	مهدي	Mehdi, Ali Mahd¯i, `Abd al-Maj¯id
Mahmud	محمود	Mehmud, Salim, 1935- Mahm¯ud, `Abd al-Ghan¯i

'u' and 'ou'



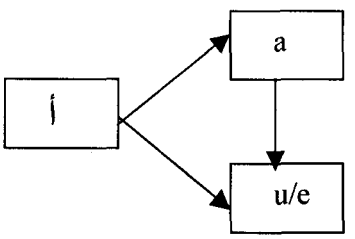
As per the LC Romanization Table أ is transliterated as 'u' and او as u̲ .So, the name م ح م د (محمّد) is transliterated as 'Muhammad' and ف وس ي(يوسف) as Yu̲suf . But in practice, two kinds of spelling variations occur while writing the Arabic name having the vowel او in English. One is that people uses 'u' to transliterate both أ and او For example 'Muhammad' and Yusuf. The second type of error occurs when people use 'ou' to transliterate Arabic vowel او instead of u̲ as indicated in LC Romanization Table. Here are some other examples in Table 5.23.

Table 5.23: Names Spelt by U or OU

Name	Name in Arabic	Name as Headings in LCNAF
Yousef	يوسف	Yusef-Zadeh, Farhad Yousef, Mohamed A. M.
Younes	يونس	Y¯unus, `Abd al-Ham¯id. Younus, Muhammad.
Sabur	صبور	Sabour, Mohammad, 1936- Sabur, Nizar, 1958-.
Nur	نور	N¯ur Ahmad Amritsar¯i, d. 1930. Nour, Mohamed A.
Mahmud	محمود	Mahmud, Abbas, 1941- Mahmoud, As`ad.

This kind of spelling variation occurs due to pronunciation differences and the influence of regional languages.

'a' and 'u/e'



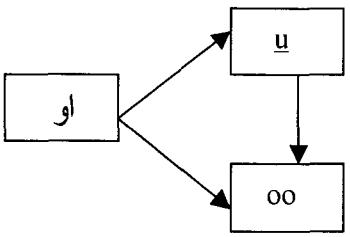
While transliterating any two Arabic letters into English, there will be a vowel to connect the two letters. For example س (s) +ف (f) = سف as saf/sef/suf , so the name يوسف is transliterated as Yusaf/Yusef/Yusuf. Two examples are given below.

Table 5.24: Names Spelt by A,U, or E

Name	Name in Arabic	Name as Headings in LCNAF
Yousuf	يوسف	Yousaf, Mohammad, 1937- Yousuf, A. R. Yousef, Mohamed A. M.
Younus	يونس	Younes, Jamal Younus, Muhammad. Younas, Malik Mohammad

This kind of spelling variation occurs due to the influence of regional languages in transliteration process. In Malayalam, the name 'Yunes' is written as യൂനസ്/യൂനുസ് When people write their name in English, they use different spellings.

'u' and 'oo'

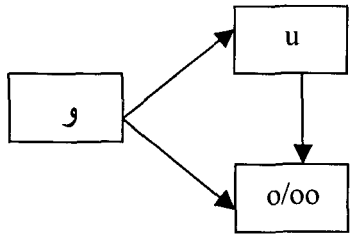


This kind of spelling variation occurs while transliterating the Arabic vowel او . As per the LC Romanization Table, او is Romanised as 'u'. But people uses 'oo' to represent او. Some examples are given below:

Table 5.25: Names Spelt by U or OO

Name	Name in Arabic	Name as Headings in LCNAF
Faruq	فاروق	F¯ar¯uq, `Abd al-Kh¯aliq Farooq, Abdul Aziz
Maimuna	مي مو نه	Maimunah Ismail Maimoona, Begum, 1933-
Mahmud	محمود	Mahmood, Akhtar Mahmud, Abbas, 1941-
Mahbub	محبوب	Mahbub, A. Q. M. Mahboob Ahmad
Mahfuz	محفوز	Mahf¯uz, `Abd al-Mun`im Ali Mahfooz

O and OO



In few names, double O is used instead of single O. For example,

Table 5.26: Name Spelt by O or OO

Name	Name in Arabic	Name as Headings in LCNAF
Mansoor	منصور	Mansor Ahmad Saman, 1949- Mansoor, Ali M.
Ghafoor	غفور	Abdul Gafoor, A. L. M. Ghafor, Abdulla

Conclusion

The analysis shows that following pairs of alphabets are simultaneously used in transliterated Arabic names.

Table 5.27: Pairs of Letters Simultaneously Used in Romanised Arabic Names

Arabic Alphabet	Romanisation by LC	Romanisation in Use
ق	Q	K
ز	Z	S
ش	SH	S
ط	T	TH
ج	J	G
ف	F	PH
ق	Q	KH
خ	KH	K
غ	GH	G
و	W	V
س	S	SS
م	M	MM
ي	Y	YY
إ	i	e
إ	i	a
ع	u	o
ي	i	ee
ي	i	ie
أ	u	o
ي	i	y
أ	a	e
او	u	ou
أ	a	u/e
او	u	u/oo

Chapter 6

A REVIEW OF OPTIONS FOR RETRIEVAL OF TRANSLITERATED ARABIC NAMES IN MAJOR OPACs

A review of options for retrieval of transliterated Arabic names in major OPACs will help to understand the importance of authority control in information retrieval. The provisions of authority control in OPACs are depended on the policy of library and options in the concerned software. Now a number of commercial software are available in the market.

The review is based on Library of Congress Name Authority File (LCNAF), which is the largest authority file in the world. Three headings available in both libraries i.e., Library of Congress and reviewed library, were taken for the study. Each heading and its see references of LCNAF were searched in other OPACs and total hits were analyzed. The options for retrieval of transliterated Arabic names in major OPACs will help to understand how the problems in name searching are solved in OPACs. The major options for name searching in OPAC are given below.

1) *See References* (4xx in MARC). Some OPAC automatically links to variant forms of heading by using See Reference as provided in the LCNAF. An example is given below:

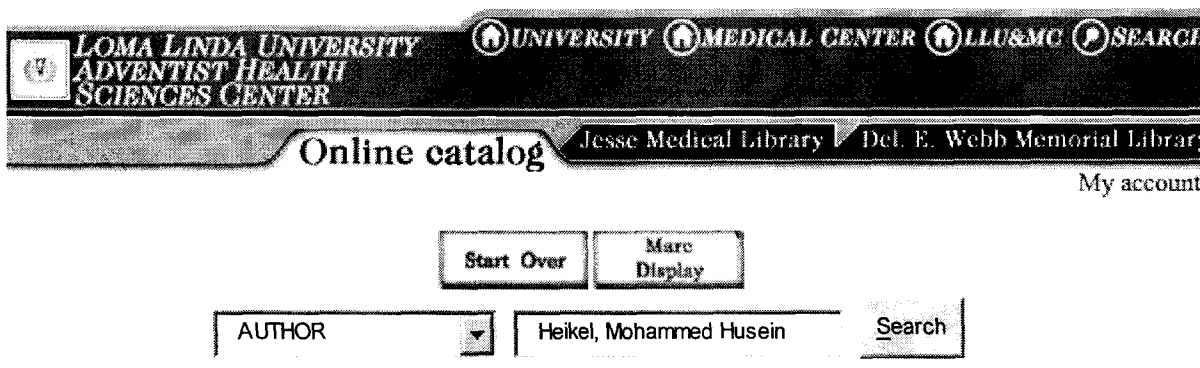


Figure 6.1: OPAC of Lomia Linda University (http://catalog.llu.edu)

2) *Inverted Heading*: Inverted heading are used to solve the problems in name searching occurred by the structure of name. A personal name may consists various elements denoting their family name, house name, place name, honorific titles etc. The structure and rendering of names have no relevance in an automated system, where the system automatically inverts the headings. An example is given below:

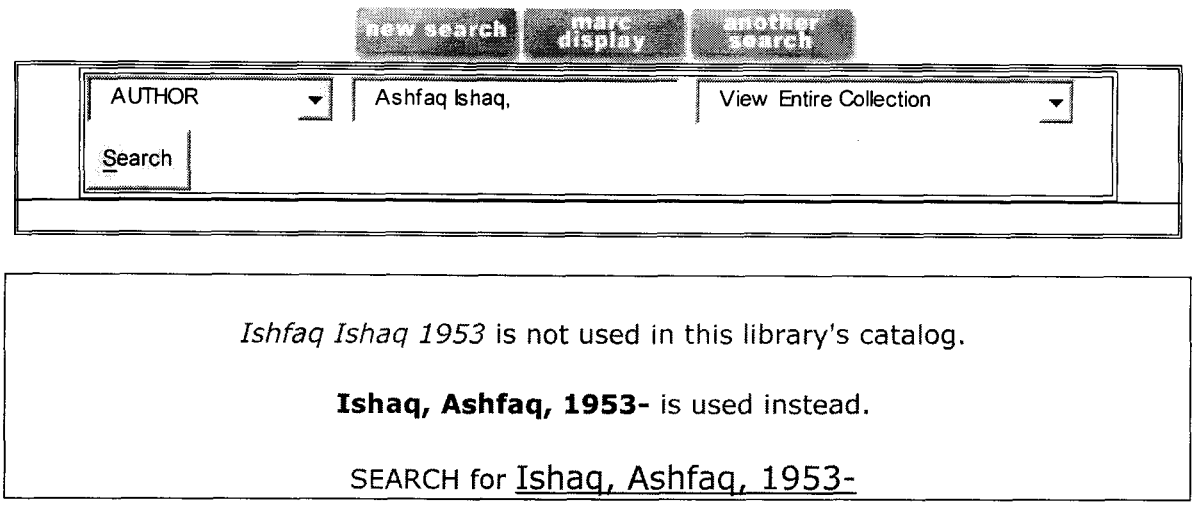


Figure.6.2: OPAC of University of South Africa http://oasis.unisa.ac.za

3) *Nearby Authors*: Some OPACs provides the option to display nearby authors in order to solve the problem of spelling variations in input. For example,

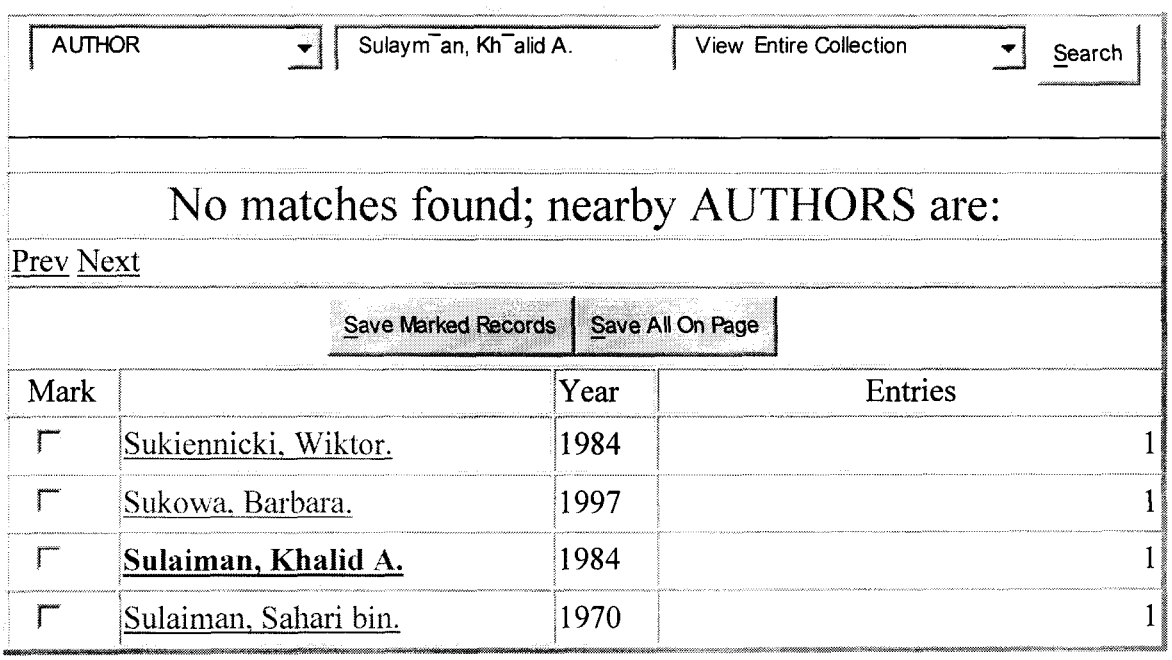


Figure.6.3: OPAC of Colorado State University http://catalog.library.colostate.edu/

In this Chapter 15, major OPACs were reviewed. The comparison was carried out in October 2004. The details are given below:

1. COPAC

COPAC is a union catalogue. It provides free access to the merged online catalogues of 24 of the largest University research libraries in the UK and Ireland, plus the British Library & the National Library of Scotland. Table 6.1 shows the comparison of Arabic name searching in COPAC and LC Authority Name Authority File.

Table 6.1: Comparison of Arabic Name Searching in LCNAF and COPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in COPAC	Nature of entry In COPAC
Haykal, Muhammad Husayn, 1888-1956	Heading	26	Heading
Haikal, Muhammad Husain, 1888-1956	See Ref.	0	No reference
Heikel, Mohammed Husein, 1888-1956	See Ref	0	No .reference
Heikel, Mohamed H 1888-1956	See Ref.	0	No.reference
Basheer, Vaikom Muhammad, 1910-	Heading	21	Heading
Vaikom Muhammad Basheer, 1910-	See Ref.	19	Inverted
Muhammad Bas¯ir, Vaikkam, 1910	See Ref.	0	No reference
Vaikkam Muhammad Bas¯ir, 1910-	See Ref.	0	No reference
Nasir Ali Mir	Heading	2	Heading
Mir Nasir Ali	See Ref.	2	Inverted
Meer Nasir Ali	See Ref.	0	No reference

The COPAC has not given See References and authority file. Hence, while search for 'Haykal, Muhammad Husayn, 1888-1956 'retrieved 26 results, Haikal, Muhammad Husain, 1888-1956 retrieved zero result. COPAC supports inverted headings. For example, while Basheer, Vaikom Muhammad, 1910 retrieved 21 hits, the inverted heading Vaikom Muhammad Basheer, 1910 retrieved 19 hits.

2. Duke University , USA

Table 6.2 shows the comparison of Arabic name searching in Duke University OPAC and LC Name Authority File.

Table 6.2: Comparison of Arabic Name Searching in LCNAF and Duke University OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in Duke Uni.	Nature of entry in Duke Uni.
R¯ashid, Rushd¯I	Heading	2	Heading
Rushd¯i R¯ashid	See Ref.	0	No Reference
Rashed, Roshdi	See Ref.	0	No Reference
Husain, `¯Amir Liy¯aqat	Heading	1	Heading
Amir Liy¯aqat Husain	See Ref.	0	Inverted
Hussain, Aamer Liaquat	See Ref.	0	No Reference
Latif, Syed Abdul	Heading	1	Heading
Syed Abdul Latif	See Ref.	0	Inverted
Latif, Sayyid Abdul	See Ref.	1	Nearby Authors

The Duke University library OPAC has not given See references. It supports inverted headings. There is option for nearby authors.

3. Edith Cowan University

Edith Cowan University (ECU) is located in Perth, Western Australia. The university uses INNOPAC software in Library (http://library.ecu.edu.au/). Table 6.3 shows the comparison of Arabic name searching in Edith Cowan University OPAC and LC Name Authority File.

Table 6.3: Comparison of Arabic Name Searching in LCNAF and ECU OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in Edith Cowan	Nature of entry in Edith Cowan Uni.
Salim Agha, Syed	Heading	1	Heading
Agha, Syed Salim	See Ref.	0	No Reference
Syed Salim Agha	See Ref.	0	No Reference
Ahmad Mansoor	Heading	1	Heading
Mansoor Ahmad	See Ref.	1	Inverted
Mohamed Yusoff Ismail.	Heading	1	Heading
Ismail, Mohamed Yusoff	See Ref.	0	No Reference

Edith Cowan University OPAC did not support See Reference entry. It support inverted heading. For example, while Ahmad Mansoor retrieved one hit, the inverted heading Mansoor Ahmad retrieved one hit.

4. University of Ottawa

The University of Ottawa is located at Canada. It is the one of the Canada's top research universities. (http://www.biblio.uottawa.ca/orbis-e.php) Table 6.4 shows the comparison of Arabic name searching in University of Ottawa OPAC and LC Name Authority File.

Table 6.4: Comparison of Arabic name searching in LCNAF and Ottawa University OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in Uni. Ottawa	Nature of Entry in Ottawa Uni. OPAC
Hassan Ibrahim	Heading	1	Nearby authors
Ibrahim, Hassan	See Ref.	1	Heading
Ibrahim, Datuk Haji Hassan	See .Ref	1	Related heading
Yousef, Yousef A.	Heading	1	Heading
Y¯usuf, Y¯usuf A.	See Ref.	0	No reference
Yousef A. Yousef	See .Ref	1	Nearby authors
Mahm¯ud, Q¯asim	Heading	1	Inverted heading
Q¯asim Mahm¯ud	See Ref.	1	Heading

The University of Ottawa library has not given See Reference entry. This OPAC retrieve inverted headings and nearby authors' list.

5. Loma Linda University

Loma Linda University (LLU) is a Seventh-day Adventist educational health-sciences institution with 3,000 students located in Southern California. (http://catalog.llu.edu/). The following table shows the comparison of Arabic name searching in Loma Linda University OPAC and LC Name Authority File.

Table 6.5: Comparison of Arabic Name Searching in LCNAF and Loma Linda University OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in Loma Linda Uni. OPAC	Nature of entry LLU. OPAC
Alhazen, 965-1039	Heading	1	Heading
Muhammad ibn al-Hasan Alh¯azin, 965-1039	See Ref.	1	Nearby Auth.
Hasan ibn Hasan ibn al-Haytham, |d 965-1039	See Ref.	1	See Ref
Ibn al-Haytham, Hasan ibn Hasan, |d 965-1039	See Ref.	1	See Ref...
Ahmed, M. Samir	Heading	1	Heading
Ahmed, Samir	See Ref.	1	Nearby Auth.
Ahmed, Mahmoud Samir	See .Ref	1	See Ref.
Haykal, Muhammad Husayn, 1888-1956	Heading	1	Heading
Haikal, Muhammad Husain, |d 1888-1956	See Ref.	1	See Ref.
Heikel, Mohammed Husein, |d 1888-1956	See .Ref	1	See Ref.

The Loma Linda University OPAC supports authority file and See Ref.erences. It also retrieves nearby headings.

6. University of Wales

Founded in 1920, the University stands in parkland overlooking Swansea Bay on the edge of the Gower Peninsula, Britain's first 'Area of Outstanding Natural Beauty' (http://library.bangor.ac.uk/). Table 6.6 shows the comparison of Arabic name searching in LCNAF and University of Wales OPAC.

Table 6.6: Comparison of Arabic name searching in LCNAF and University of Wales OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in UW OPAC	Nature of Entry in University of Wales
Jinnah, Mahomed Ali, 1876-1948	Heading	1	Heading
Muhammad `Al¯i Jinnah, 1876-1948	See Ref.	1	See Ref.
Quaid-i-Azam	See .Ref	1	See Ref.
Faruqee, Rashid, 1938-	Heading	1	Heading
Rashid Faruqee	See Ref.	1	See Ref.
Faruqee, R	See .Ref	1	See Ref.
A H. Ahmad Sarji	Heading	1	Heading
Hamid, Ahmad Sarji Abdul	See Ref.	1	See Ref.
Sarji, A. H. Ahmad	See .Ref	1	See Ref.

The University of Wales OPAC provides options for authority control and gives See References similar to Library of Congress.

7. University of St. Andrews.

St Andrews is Scotland's first University and the third oldest in the UK Founded in 1413. (http://138.251.116.3/). The following table shows the comparison Arabic name searching in University of St. Andrews OPAC and LC Name Authority File.

Table 6.7: Comparison of Arabic Name Searching in LCNAF and University of St. Andrews OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in SAULCAT	Nature of Entry in SAULCAT
Ali, Mohamed	Heading	1	See Ref.
Ali, Muhammad	See Ref.	1	Heading
Mohammad Ali, |c Maulana,	See Ref.	0	No references
Sulaiman, Khalid A.	Heading	1	Heading
Sulaym¯an, Kh¯alid A.	See Ref.	1	Nearby auth
Moustafa, Ahmed Youssef.	Heading	1	Heading
Mustaf□Ahmad Y¯usuf	See Ref.	1	See Ref.

SAULCAT partially support authority control file and provides See Ref.erence entry. It also provides nearby authors list also.

8. University of South Africa

The University of the Cape of Good Hope, renamed the University of South Africa in 1916, was created by Act 16 of 1873 of the Cape of Good Hope Parliament (http://oasis.unisa.ac.za/). Table 6.8 shows the comparison of Arabic name searching in LCNAF and University of South Africa OPAC.

Table 6.8: Comparison of Arabic Name Searching in LCNAF and University of South Africa OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in Natio. Lib of S.A	Nature of Entry in National. Library of South Africa
Muhammad Zakariyy¯a	Heading	1	Heading
Zakariya, Muhammad	See Ref.	1	Nearby auth.
Zakariyy¯a, Muhammad	See Ref.	1	Inverted heading
Rasheed, Sadig	Heading	1	Heading
Rash¯id, Sad¯iq	See Ref.	1	See Ref.
Rasheed, Sadiq	See Ref.	1	See Ref.
Ishaq Ashfaq	Heading	1	Heading
Ashfaq Ishaq,	See Ref.	1	Inverted Heading

University of South Africa OPAC support See References, inverted heading and nearby authors list.

9. Colorado State University

Colorado State University is the one of the leading institutions in Colorado State, USA(http://catalog.library.colostate.edu/). The Table 6.9 shows the comparison of Arabic name searching Colorado State University OPAC and LC Name Authority File.

Table 6.9: Comparison of Arabic Name Searching in LCNAF and Colorado University OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in Colorado State UL	Nature of Entry in Co. State Uni. Library
Salam, Abdus, 1926-	Heading	1	Heading
Muhammad `Abd al-Sal¯am, |**d** 1926-	See Ref.	1	See Ref.
Salam, Muhammad Abdus, |**d** 1926-	See Ref.	1	See Ref.
Sulaiman, Khalid A	Heading	1	Heading
Khalid A. Sulaiman	See Ref.	1	See Ref.
Kh¯alid A. Sulaym¯an	See Ref.	1	See Ref.
Ashraf, Mohammad	Heading	1	Heading
Ashraf, Kazi Mohammad	See Ref.	1	See Ref.

Colorado State University OPAC provides See Reference entry as Library of Congress.

10. University of Essex

The University of Essex, based at Wivenhoe Park, Colchester, received its Royal Charter in 1966. (http://serlib0.essex.ac.uk/). The following table shows the comparison of Arabic name searching in LCNAF and University of Essex OPAC.

Table 6.10: Comparison of Arabic Name Searching in LCNAF and University of Essex

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in Essex	Nature of Entry in Essex OPAC
Basheer, Tahseen	Heading	1	Heading
Bashir, Tahseen	See Ref	0	Not found
Tahseen Basheer	See Ref	1	Inverted reference
Ahmad Ibrahim	Heading	1	Nearby Authors
Ahmad Mohamed Ibrahim	See Ref	0	Not found
Ahmad bin Mohamed Ibrahim	See Ref	1	Heading
Ahmad, Imtiaz, 1940	Heading	1	Heading
Imtiaz Ahmad	See Ref	1	Inverted reference

University of Essex OPAC does not provide the provision for See Reference. However, it support inverted heading.

11. Ohio University

Ohio University is one of the Universities in Greece, situated at Athens. (http://www.library.ohiou.edu/). The following table shows the comparison of Arabic name searching in LCNAF and Ohio University OPAC.

Table 6.11: Comparison of Arabic name searching in LCNAF and Ohio University OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in Ohio	Nature of Entry in Ohio
Bin Laden, Osama	Heading	1	Heading
Usama bin Laden	See Ref.	1	See Ref.
Ibn L¯adin, Us¯amah, |	See Ref.	1	Inverted reference
Zakaria bin Haji Ahmad	Heading	1	Heading
Ahmad, Zakaria bin Haji,	See Ref.	1	See Ref.
Zakaria Haji Ahmad	See Ref.	1	See Ref.
Amirsadeghi, Hossein	Heading	1	Heading
S¯adiq¯i, Husayn Am¯ir	See Ref.	1	See Ref.

Ohio University supports See Reference entry and inverted reference.

12. Oxford Library and Information System NB 4689

The OLIS web OPAC is the web interface which enables searching in Oxford University's online union library catalogue (http://www.lib.ox.ac.uk/olis/). Table 6.12 shows the comparisions of Arabic names searching in OLIS and LCNAF.

Table 6.12: Comparison of Arabic name searching in LCNAF and OLIS OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in OLIS	Nature of Entry in OLIS
B¯ar¯ud¯i, `Abd All¯ah `Umar	Heading	1	Heading
Abd All¯ah `Umar al-B¯ar¯ud¯I	See Ref.	1	See Ref.
Husayn¯i, `Abd All¯ah ibn `Umar al-B¯ar¯ud¯i	See Ref.	0	Not found
Bashier, Zakaria.	Heading	1	Heading
Zakaria Bashier	See Ref.	1	Inverted reference
Bash¯ir, Zakar¯iy¯a	See Ref.	0	Not found
Nadv¯i, Sayyid Sulaim¯an	Heading	1	Heading
Syed Sulaiman Nadvi	See Ref.	0	Not found
Syed Suleyman Nadvi,	See Ref.	0	Not found

OLIS does not support See Reference.

13. The Austrian National University, Canberra (http://library.anu.edu.au/search~S1/).

The following table shows the comparison of Arabic name searching in LCNAF and Australian National University OPAC.

Table 6.13: Comparison of Arabic name searching in LCNAF and ANU OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in ANU	Nature of Entry in ANU
Abdesselem, Mohamed	Heading	1	Heading
Mohamed Abdesselem	See Ref.	1	See Ref.
Muhammad `Abd al-Sal¯am	See Ref.	1	See Ref.
Ish¯aq, Muhammad Qamar, 1961-	Heading	1	Heading
Ishaque, Mohammad Qamar,1961-	See Ref.	1	See Ref.
Mohammad Qamar Ishaque, 1961	See Ref.	1	See Ref.
Yousef, Mohamed K.	Heading	1	Heading
Yousef, M. K	See Ref.	1	Nearby Authors

Australian National University headings and references are similar to Library of Congress.

14. Trinity Theological College, Australia (http://www.trinity.qld.edu.au/)

Table 6.14 shows the comparisions of arabic names searching in Trinity Theological College OPAC and LCNAF.

Table 6.14: Comparison of Arabic name searching in LCNAF and Trinity OPAC

Entry in LC Authority Record	Nature of Entry in LC	Hits in Trinity OPAC	Nature of Entry in Trinity Theological college
Muhammad Abul Quasem.	Heading	1	Heading
M. A. Quasem	See Ref.	0	Not found
Abul Quasem	See Ref.	1	Heading
Ahmad, Bashiruddin Mahmud,	Heading	1	Heading
Bashiruddin Mahmud Ahmad	See Ref.	1	Heading
Mahmood Ahmad, Bashir-ud-Din	See Ref.	0	Not found
Nasr, Seyyed Hossein	Heading	12	Heading
Nasr, Sayyid Husayn	See Ref.	0	Not found
Nasr, Hossein	See Ref.	12	Heading

Trinity Theological College does not support See Reference.

15. University of Nevada, Rino (http://www.library.unr.edu/)

The following table shows the comparison of Arabic name searching in LCNAF and University of Nevada OPAC.

Table 6.15: Comparison of Arabic name searching in LCNAF and University of Navada OPAC

Entry in LC Name Authority Record	Nature of Entry in LC	Hits in UNL OPAC	Nature of Entry in University of Nevada Library
Badaw¯i, Muhammad Mustafá	Heading	1	Heading
Badaw¯i, Mustafá	See Ref.	1	See Ref.
Badawi, M. M.	See Ref.	1	See Ref.
Hussain, Asaf	Heading	1	Heading
Asaf Hussain	See Ref.	1	Inverted Heading
Hüseyin, Asaf	See Ref.	0	Not found
Kassim, Namir E.	Heading	1	Heading
Q¯asim, Nam¯ir al-Shaykh	See Ref.	1	See Ref.

University of Nevada OPAC supports See Reference and inverted headings.

Analysis

The major 15 OPACs were compared on basis of its retrieval effectiveness in Arabic name searching. Table 6.16 shows the comparison of major OPACs

Table 6.16: Comparison of Options of Name Searching in Major OPACs

SL. No	Name of the University/OPAC	Heading as in LCNAF	Inverted Reference	See References	Nearby Author's
1	COPAC	Yes	Yes	No	No
2	Duke	Yes	Yes	No	Yes
3	Edith Cowan	Yes	Yes	No	No
4	Ottawa	Yes	No	No	Yes
5	Lona Linda	Yes	No	Yes	Yes
6	Wales	Yes	No	Yes	No
7	St.Andrew	Yes	No	Yes	Yes
8	South Africa	Yes	Yes	Yes	Yes
9	Colorado	Yes	No	Yes	No
10	Essex	Yes	Yes	No	Yes
11	Ohio	Yes	Yes	Yes	No
12	Oxford	Yes	Yes	Yes	No
13	Trinity	Yes	No	No	No
14	Australia	Yes	Yes	Yes	Yes
15	Nevada	Yes	Yes	Yes	No

The analysis shows following findings:

1. All headings are similar to Library of Congress headings
2. Most OPACs have the option of Inverted Reference (i.e. Inverted Heading)
3. Most OPACs provide See References.
4 Only 6 OPACs have option of Nearby Authors.

Reference

1. Veerankutty Chelatayakkot and V. Jalaja " *Effectiveness of Name Searching in Web OPAC : From Authority Control to Access Control*" in Proceedings of the 3rd International CALIBER , Ahmadabad , Inflibnet, 2005 p348-357.

Chapter 7

CONCLUSION

Automated Authority Control System

The proposed study is intended to understand the variant spellings of Arabic name in usage and the influence of regional languages in transliteration process. This will help to propose an new authority control system in which the system automatically creates the major variant spellings of a unique name and retrieve the related results. The proposed authority control system would not be based on any author who is using different names, but it would be based on names and variant spellings in usage.

Generally, in all automated transliteration systems, the input (i.e. source language such as Arabic, Chinese) is transliterated into target language (i.e. most probably English). But in this study, the researcher considered the transliterated Arabic names and its variant spellings used by the people. In the proposed system, the input is transliterated Arabic name. The system automatically would create major variant spelling of the input and will search in the connected database and retrieve the results.

The variant spellings of the input name are created by permuting the alphabets, which are obtained from the analysis of transliterated Arabic names. The following table shows the pairs of alphabets, which are found as simultaneously used in transliterated Arabic names.

Table 7.1: Roman Alphabets Commonly Found in Transliterated Arabic Names

Consonants	Double letters	Vowels
Q ◄---► K	S◄---►SS	i◄---►e
S ◄---► Z	M◄---►MM	i◄---►a
S ◄---► SH	O◄---►OO	i◄---►ee
T ◄---►TH	Y◄---►YY	i◄---►y
J ◄---► G		i◄---►ie
F ◄---► PH		a◄---►e
Q ◄---► KH		u◄---►ou
K ◄---► KH		u◄---►oo
V ◄---► W		o◄---►u

By permutation of the pairs of letters shown above, the possible numbers of variant spellings of a unique Arabic name can be created. But the afore said pairs of letter may not permute always in all names having the same letter. For example,

when ***i*** *is permuted as* ***ee,*** *then Bash****i****r is permuted as Bash****ee****r which is correct, but when Za****i****naba is permuted as Za****ee****naba, it is wrong.*

So, while manipulating the common variant alphabet, the system will retrieve false hits. In order to overcome this problem, a filter (program) has to be incorporated in the system.

In any automated transliteration system, the input is given in the source language and the system can easily understand it, especially in the case of vowels. But, in the proposed system, the input is transliterated name and so we have to create the conditions of vowels permutation (filter). So, the preparation of algorithms for filter is a tedious task.

Here is an example: Input name ASHREF

Pairs of alphabet to be permuted: A ◄---►E , S◄---►SH, S◄---►Z

Possible Permutations Ashref Asref Asraf Azraf Azhref , Esraf, Eshref and more..

After Filtering Ashref Ashraf Asref and Asraf

A search in the connected database and the system will retrieve the hits found in the database. The proposed system can be diagrammatically represented as follows:

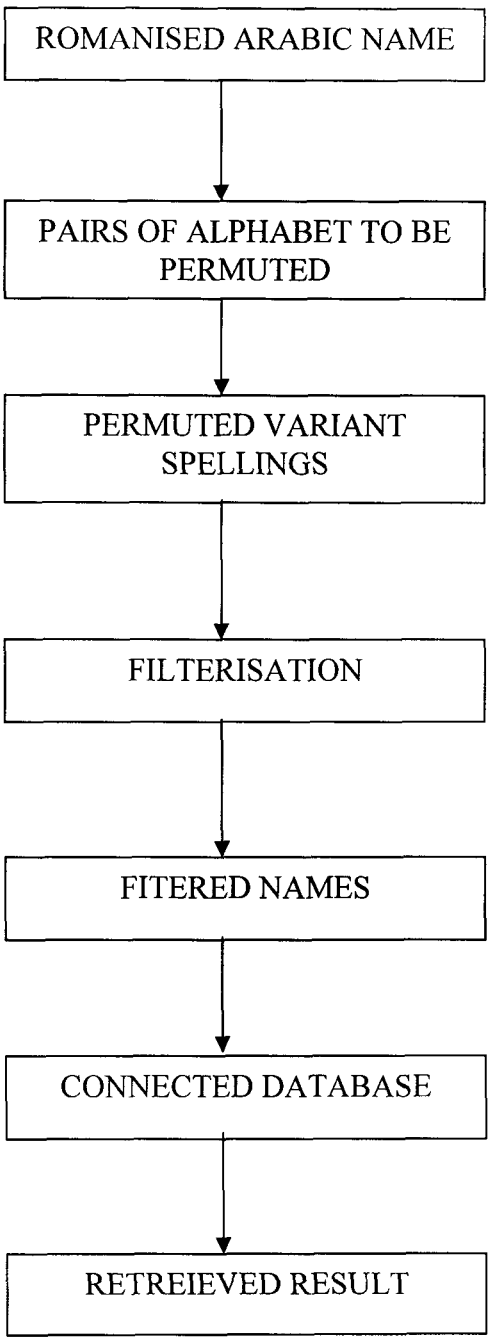


Figure 7.1: Diagram of Automated Authority Control System

The Filter Program

The filter program is written in C language with a user interface in Visual Basic 6. The program is available in the accompanying CD-ROM. Also, source code is attached as Appendix –II.

Performance Evaluation of the Program

The performance of the program was evaluated by comparing common variant spellings of Arabic names in use and output of the program. *Google* search engine is used for this purpose. Common spelling permutations were checked.

Methodology: Each name's possible (not rare) spelling variations were searched in *Google* search engine and total hits were summed. This sum was taken as 100. Then, the percentage of each spelling variations were calculated. By using this percentage for each spelling variations, the performance of the output were calculated. The performance evaluation was done on March 2005. Since *Google* search engine indexes thousands of new web pages daily, the figure may be changed.

The program shows 83.2% performance in creating common spelling variations used by the people. The details of performance evaluation are given in Table 7.2.

Table 7.2: Performance Evaluation of the Automated Authority Control System

Name/ Input (1)	Permu tation (2)	Commonly Found Variant Spellings (3)	Hits in Google (4)	Per Cent (Total of (4)) as Hundred) (5)	Output (6)	Per Cent (7)	Total Of (7) (8)
Ahmad	a/e	Ahmad Ahmed Ahammad Ahammed	2,570,000 4,370,000 4,340 781	37.00 62.92 0.06 0.02	Ahmad Ahmed	37.00 62.92	99.92
Ahmed	e/a	---do--	---do--	---do--	---do—	---do--	99.92
Gafoor	oo/ou oo/o	Gafoor Gafour Gafur Gafor	19,800 2,560 15,500 8,540	42.67 5.52 33.41 18.40	Gafoor Gafour Gafur	42.67 5.52 33.41	81.6
Gafur	u/ou	---do--	---do--	---do--	Gafoor Gafour Gafur	42.67 5.52 33.41	81.6
Gafor	o/oo	---do--	---do---	---do--	Gafor	18.40	18.40
Gafour	ou/u	---do--	---do---	---do--	Gafoor Gafour Gafur	42.67 5.52 33.41	81.6

Name/ Input (1)	Permu tation (2)	Commonly Found Variant Spellings (3)	Hits in Google (4)	Per Cent (Total of (4)) as Hundred) (5)	Output (6)	Per Cent (7)	Total Of (7) (8)
Raheem	ee/i ee/ie ee/i	Raheem Rahim Rahiem Rehim Rehiem Reheem	191,000 670,000 9,570 12,700 139 575	21.60 75.80 1.08 1.45 .01 .06	Raheem Rahim Rahiem	21.60 75.80 1.08	98.48
Rahim	i/ee i/ie	--do-	- --do--	---do—	Raheem Rahim Rahiem	21.60 75.80 1.08	98.48
Rauf	u/oo	Rauf Raouf Raoof	589,000 120,000 9,970	81.92 16.69 1.39	Rauf Raouf Raoof	81.92 16.69 1.39	100
Raouf	ou/oo	--do-	--do-	--do-	--do-	--do-	100
Raoof	oo/u	--do-	--do-	--do-	--do-	--do-	100
Ashraf	sh/s a/e	Ashraf Ashref Asraf Asref	426,000 1,020 9,020 960	97.48 0.24 2.06 0.22	Ashraf Ashref Asraf Asref	97.48 0.24 2.06 0.22	100

Name/ Input (1)	Permu tation (2)	Commonly Found Variant Spellings (3)	Hits in Google (4)	Per Cent (Total of (4)) as Hundred) (5)	Output (6)	Per Cent (7)	Total Of (7) (8)
Ashref	sh/s e/a	Ashraf Ashref Asraf Asref	426,000 1,020 9,020 960	97.48 0.24 2.06 0.22	Ashraf Ashref Asraf Asref	97.48 0.24 2.06 0.22	100
Asraf	s/sh	--do-	--do-	--do-	Asraf Asref	2.06 0.22	2.28
Anwar	w/v	Anwar Anvar Anwer Anver	1,040,000 210,000 85,300 102,000	72.36 14.61 5.94 7.09	Anwar Anvar Anwer Anver	72.36 14.61 5.94 7.09	100
Anvar	v/w	--do-	--do-	--do-	Anwar Anvar Anwer Anver	72.36 14.61 5.94 7.09	100
Anver	v/w e/a	--do-	--do-	--do-	Anwar Anvar Anwer Anver	72.36 14.61 5.94 7.09	100
Fathima	th/t	Fathima Fatima	22,500 1,36,000	14.20 85.80	Fathima Fatima	14.20 85.80	100
Fatima	t/th	--do-	--do-	--do-	Fathima Fatima Fathema	14.20 85.80	100

Name/ Input (1)	Permu tation (2)	Commonly Found Variant Spellings (3)	Hits in Google (4)	Per Cent (Total of (4)) as Hundred) (5)	Output (6)	Per Cent (7)	Total Of (7) (8)
Hassan	ss/s a/e	Hassan Hassen Hasan Hasen	3,950,000 1,490,000 1,890,000 801,000	48.58 18.32 23.24 9.86	Hassan Hassen Hasan Hasen	48.58 18.32 23.24 9.86	100
Hasan	s/ss	---do --	---do---	---do---	Hasan Hasen	23.24 9.86	33.10
Hassen	ss/s e/a	---do --	---do---	---do---	Hassan Hassen Hasan Hasen	48.58 18.32 23.24 9.86	100
Mohammad	mm/m a/e o/u	Mohammad Mohammed Mohamad Mohamed Muhammad Muhammed Muhamad Muhamed	3,430,000 5,950,000 738,000 4,870,000 4,790,000 664,000 117,000 115,000	16.59 28.78 3.57 23.55 23.17 3.21 0.57 0.56	Mohammad Mohammed Mohamad Mohamed Muhammad Muhamad	16.59 28.78 3.57 23.55 23.17 0.57	96.23
Mohammed	mm/m e/a o/u	---do --	---do---	---do---	Mohammad Mohammed Mohamad Mohamed Muhammed Muhamed	16.59 28.78 3.57 23.55 23.17 0.57	96.23

Name/ Input (1)	Permu tation (2)	Commonly Found Variant Spellings (3)	Hits in Google (4)	Per Cent (Total of (4)) as Hundred) (5)	Output (6)	Per Cent (7)	Total Of (7) (8)
Mohamed	o/u	Mohammad	3,430,000	16.59	Muhamed	0.56	
	e/a	Mohammed	5,950,000	28.78	Muhamad	0.57	28.25
		Mohamad	738,000	3.57	Mohamad	3.57	
		Mohamed	4,870,000	23.55	Mohamed	23.55	
		Muhammad	4,790,000	23.17			
		Muhammed	664,000	3.21			
		Muhamad	117,000	0.57			
		Muhamed	115,000	0.56			
Muhammad	mm/m	---do --	---do---	---do---	Muhammad	23.17	
					Muhamad	0.57	27.51
					Muhammed	3.21	
					Muhamed	0.56	
Hussain	i/y	Hussain	1,390,000	48.65	Hussain	48.65	
	s/ss	Hussayn	11,700	0.41	Hussayn	0.41	81.79
	o/u	Husain	688,000	24.08	Husain	24.08	
		Husayn	208,000	7.28	Husayn	7.28	
		Hossain	482,000	16.87	Husseyn	0.19	
		Hosain	29,200	1.02	Huseyn	1.18	
		Hosayn	9,070	0.32			
		Husseyn	5,290	0.19			
		Huseyn	33,800	1.18			
Hussayn	y/i	---do --	---do---	---do---	Hussayn	0.41	
					Husayn	7.28	80.42
					Hussain	48.65	
					Husain	24.08	

Name/ Input (1)	Permu tation (2)	Commonly Found Variant Spellings (3)	Hits in Google (4)	Per Cent (Total of (4)) as Hundred) (5)	Output (6)	Per Cent (7)	Total Of (7) (8)
Hossain	o/u	Hussain	1,390,000	48.65	Hossain	16.87	
	ss/s	Hussayn	11,700	0.41	Hosain	1.02	90.94
	i/y	Husain	688,000	24.08	Hussain	48.65	
		Husayn	208,000	7.28	Husain	24.08	
		Hossain	482,000	16.87	Hosayn	0.32	
		Hosain	29,200	1.02			
		Hosayn	9,070	0.32			
		Husseyn	5,290	0.19			
		Huseyn	33,800	1.18			
Qamar	q/kh	Qamar	202,000	86.71	Qamar	86.71	
	q/k	Khamar	11,500	4.94	Khamar	4.94	98.03
	a/e	Kamar	8,400	3.61	Kamar	3.61	
		Qamer	6,550	2.81	Qamer	2.81	
		Khamer	4,500	1.93			
Khamar	kh/q				Khamar	86.71	
	kh/k	---do --	---do---	---do---	Qamar	4.94	97.19
	a/e				Kamar	3.61	
					Khamer	1.93	
Kamar	k/q	---do --	---do---	---do---	Kamar	3.61	3.61
	k/kh				Kamer		

Name/ Input (1)	Permu tation (2)	Commonly Found Variant Spellings (3)	Hits in Google (4)	Per Cent (Total of (4)) as Hundred) (5)	Output (6)	Per Cent (7)	Total Of (7) (8)
Zaid	z/s	Zaid	248,000	0.12	Zaid	0.12	
		Said	222,000,000	99.58	Said	99.58	99.87
		Zayd	82,100	0.03	Zayd	0.03	
		Sayd	314,000	0.14	Sayd	0.14	
		Seyd	47,100	0.03			
		Saed	238,000	0.10			
Said	s/z	Zaid	248,000	0.12	Said	99.58	
		Said	222,000,000	99.58	Sayd	0.14	99.85
		Zayd	82,100	0.03	Seyd	0.03	
		Sayd	314,000	0.14	Saed	0.10	
		Seyd	47,100	0.03			
		Saed	238,000	0.10			
Osama	o/u	Osama	5,190,000	93.04	Osama	93.04	
		Usama	388,000	6.96	Usama	6.96	100
Usama	u/o	Osama	5,190,000	93.04	Osama	93.04	
		Usama	388,000	6.96	Usama	6.96	100

The evaluation of the performance of the program shows 83.38 % success in creating commonly found variant spellings of a unique Arabic name. The researcher also analysed the performance of permutation of alphabets in this program, which is given in the Table 7.3

Table 7.3: Performance of Alphabets Permutation

Sl. No	Permutation of Alphabet	Performance
(1)	(2)	(3)
1	Q as K	Satisfactory
2	K as Q	Not Satisfactory
3	S as Z	Not Satisfactory
4	Z as S	Satisfactory
5	S as SH	Not Satisfactory
6	SH as S	Satisfactory
7	T as TH	Satisfactory
8	TH as T	Satisfactory
9	J as G	Not Satisfactory
10	G as J	Not Satisfactory
11	F as PH	Not Satisfactory
12	PH as F	Not Satisfactory
13	Q as KH	Satisfactory
14	KH as Q	Satisfactory
15	V as W	Satisfactory
16	W as V	Satisfactory
17	S as SS	Not Satisfactory
18	SS as S	Satisfactory
19	M as MM	Not Satisfactory
20	MM as M	Satisfactory
21	O as OO	Not Satisfactory
22	OO as O	Not Satisfactory
23	Y as YY	Not Satisfactory
24	YY as Y	Not Satisfactory

Sl. No	Permutation of Alphabet	Performance
(1)	(2)	(3)
25	I as E	Satisfactory
26	E as I	Not Satisfactory
27	I as A	Not Satisfactory
28	A as I	Not Satisfactory
29	I as EE	Satisfactory
30	EE as I	Satisfactory
31	I as Y	Satisfactory
32	Y as I	Satisfactory
33	I as IE	Satisfactory
34	IE as I	Satisfactory
35	A as E	Satisfactory
36	E as A	Satisfactory
37	U as OU	Satisfactory
38	OU as U	Satisfactory
39	U as OO	Satisfactory
40	OO as U	Satisfactory
41	O as U	Satisfactory
42	U as O	Satisfactory
43	OU as OO	Satisfactory
44	OO as OU	Satisfactory

Table 7.3 shows that out of 44 permutations, 28 pairs (63.64%) permute satisfactorily and 16 pairs (36.36%) does not permute. Among these 16 pairs, four pairs i.e., J as G, G as J, PH as F and F as PH are rarely used in names in Egypt and some African countries. However, cent percent permutation has no significance, where it will produce wrong names.

Application of Automated Name Authority Control System

This automation program has wider applications in Information Retrieval.

1. In OPAC, it helps to replace the current practice of authority control. It supports access control, the "super authority record". While authority control record declare a heading as "authorized" form, access control record links all variations without declaring one heading as authorized form. The painstaking cataloguing rules for how to construct authorized forms can be put off. The cost of authority work can be minimized.

2. This program can be used as an interface in database name searching. When a searcher key in a known form (spelling) of a name, the system automatically creates spelling variations and displays the requested item even though the preferred spelling might be quite different from what is entered The researcher has developed a prototype interface for this purpose.

3. This program can be used as a tool in free text name searching in corpora like newspaper articles, press cuttings etc.

4. This program helps to detect Arabic proper names and their spelling variations in automatic speech recognition output, where, the pronunciation difference, in speech recognition system, creates spelling variations.

5. This program will be helpful in detecting spelling errors, omissions, and deletions of Arabic names in databases.

6. This program can be used as a tool to increase recall in Information Retrieval.

BIBLIOGRAPHY

BIBLIOGRAPHY

ALA Glossary of Library and Information Science, Chicago, ALA, 1985.

Abduoulay, Kaba. "Perception of Cataloguers and End User Towards Bilingual Authority File" *The Electronic Library* 20 (2002): 202-210.

Arbabi, M. *et al* "Algorithms for Arabic Name Transliteration." *IBM Journal of Research and Development* 38 (1994): 183-205.

Ayres, FH. "Authority Control Simply Does Not Work" *Cataloging and Classification Quarterly* .32.2(2001): 49-59.

Banerjee, Kyle. "Describing Remote Electronics Documents in the Online Catalogue: Current Issues" *Cataloging and Classification Quarterly* 25.1 (1997): 5-20.

Barnhart, Linda. 1996. "Access Control Records: Precepts and Challenges" *Authority Control in 21st Century: An invitational Conference* 10March 2005 <http://digitalarchive.oclc.org/>

Beheshti, Jamshid. "The evolving OPAC" *Cataloging and Classification Quarterly* 24.1/2 (1997):163-185.

Boese, Kent C. "What in a Name: Associated Cost of Authority Work for Artist Names: the Bottom Line" *Managing Library Finance* 16.3 (2003): 106-110.

Burger, Robert H. A*uthority Work : the Creation, Use, Maintenance and Evaluation of Authority Records and Files.* Littleton, Colo.:Libraries Unlimited,1985

Calhoun, Karen.1998."A Birds Eye View of Authority Control in Cataloguing", *Proceedings of the Taxonomic Authority File Workshop* 10 March 2005 <http://www.calacademy.org/research/informatics/taf/proceedings/Calhoun.html >

Cutter C.A. *Rules for a Dictionary Catalogue.* 4th ed. Washington: Government Printing Office, 1904.12

Encyclopedia of Islam Ed. Evan Donzel et al. New ed. Vol. 4. Leiden: E.J.Brill, 1978. 179-181

Fiander, David J. "Applying XML to the Bibliographic Description." *Cataloging and Classification Quarterly* 33.2 (2001):17-28.

Fox, Judith A and Kay Kanafani, "Global Chang Capabilities to Improve Authority Control in an Online Catalogue." *Information Technology and Library* 8(1989): 273-283.

Gentile–Tedeschi, Massino and Federica. Riva, 2003. " Authority Control in the Field of Music: Names and Titles" *Proceedings of International Conference on Authority Control* 10March 2005 http://www.unifi.it/universita/biblioteche/ac/relazioni/gentili-tedeschi_eng.pdf>

Hong, Yoojin, Byung-Won On and Dongwon Lee. "System Support for Name Authority Control Problem in Digital Libraries: OpenDBLP Approach" in *Lecture Notes in Computer Science*: Springer-Verlag, 3232(2004): 134-144.

International Conference on Cataloguing Principles. *Report.* Paris :ICCP, 1961.

Jabraj, V.F.D. "Online Public Access Catalogue" *Indian Journal of Information Library and Society* (2003) :147.

Jeong, Kil Soon *et al.* "Automatic Identification and Back-transliteration of Foreign Words of Information Retrieval." *Information Processing and Management.* 35 (1999): 523-540.

Johnson, Bruce Chr. " XML and MARC: Which is "Right"?" *Cataloging and Classification Quarterly* 32.1 (2001): 81-90.

Khurshid, Zahiruddin. "Arabic Script Materials: Cataloging Issues and Problems" *Cataloging and Classification Quarterly* 34.4(2002) 67-77.

---------- " From MARC to MARC21 and Beyond: Some Reflections on MARC and the Arabic Language" *Library Hi Tech* 20 (2002): 370-377.

Ki-Tat Lam(2002) "XML and Global Name Access Control" *OCLC System Services* 18(2) 88-96.

Kock, Gideon de V. de "Searching on Full Name Providing for Spelling Variations" in *Proceedings of the 2002 Annual Research Conference of the South African Institute for Computer Scientists and Information Technologists on Enablement Through Technology*, 2002 .255-255.

Krieger, Mitchael T. "Characteristics of the 670 Field in Records for Names in the Anglo-American Authority File." *Cataloging and Classification Quarterly* 23.1 (1996): 99-119.

Lam, Vinth-The. "Outsourcing Authority Control: Experience of the University of Saskatchewan Libraries." *Cataloging and Classification Quarterly*.32.4 (2001) 53-69.

Larkey,Leah S, Nasreen Abdul Jaleel and,Margaret Connell. 2003."What is in a Name?: Proper Names in Arabic Cross Language Information Retrieval " *CIIR Technical Report IR 278* 10 March 2005 <http://ciir.cs.umass.edu/publications/>

Lewellen, Mark. 1999."Name Searching with Artificial Neural Network" *Workshop on Natural Language Processing and Neural Networks* 10March 2005 <http://www.math.ryukoku.ac.jp/~qma/activity/NLPNN99/>

Ling Hwey Jang. "What Authority? What Control?" *Cataloguing and Classification Quarterly* 34.4 (2002): 91-97.

Mansor, Yushiana. "Bibliographic Exchange in Malaysia: Variations in Name Headings" *Library Review* 52.1(2003): 38-42.

Matters, Marion. "Authority Work for Transitional Catalogues." *Cataloging and Classification Quarterly*. 11(1990): 53-69.

Nasreen, Abdul Jaleel and Leah S. Larkey 2003. "Statistical Transliteration for English-Arabic Cross Language Information Retrieval" *Proceedings of the Twelfth International Conference on Information and Knowledge Management* 10March 2005 <http://ciir.cs.umass.edu/pubfiles/ir-293.pdf>

---------- and. Leah S Larkey 2002. "English to Arabic Transliteration for Information Retrieval : A Statistical Approach" *CIIR Technical Report* 10 March 2005 <http://ciir.cs.umass.edu/pubfiles/ir-261.pdf>

Olson, Chalermsee. "Cataloging Southeast Asian Language Materials: the Case of the Thai Languages" *Cataloging and Classification Quarterly* 22.2(1996): 19 – 28.

Pappas, Evan. "An Analysis of Eight RLIN Member's Authority Controlled Access Points for Purpose of Speeding Copy Catalogue Workflow" *Cataloging and Classification Quarterly* 22.1(1996): 29- 47.

Pfeifer, Olrich, Thomas Poersch and Norbert. Fuhr "Retrieval Effectiveness of Proper Name Search Methods" *Information Processing and Management* 32(1996): 667-679.

Plettner, Martha Sceirs 2003. "Arabic Name Authority in the Online Environment: Options and Implication" 15 March. 2005 <http://www.uni-bamberg.de/unibib/melcom/PlettnerICBC.html>

Raghavan, Hema and James Allan 2004. "Proper Names and Their Spelling Variations in Automated Speech Recognition Output" 10 March 2005 <http://ciir.cs.umass.edu/pubfiles/ir-361.pdf>

Ranganathan, S.R. *Classified Catalogue Code with Additional Rules for Dictionary Catalogue Code.* 5th ed. Bombay: Asia, 1964.155.

Ruiz-Perez, R. "Consequence of Applying Cataloging Codes for Author Entries to the Spanish Library Online Catalogue" *Cataloguing and Classification Quarterly* 32.2 (2001): 31-35.

Snyman,MMM and Rensberg, Jansen Van. "Reengineering Name Authority Control *"The Electronic Library* 17 (1999):.313-322.

Spink,Amanda and Bernard J. Jansen. "Searching for People on the Web Search Engines" *Journal of Documentation* 60(2004):266-278.

Sridhar M.S. "OPAC Vs Card Catalogue: A Comparative Study of User Behavior" *The Electronic Library* 22(2004): 175-183.

Stalls, Bonnie Glover, and knight, Kevin 1998. "Translating Names and Technical Terms in Arabic Text" 10 March 2005 <http://acl.ldc.upenn.edu/ W/W98/W98-1005.pdf>

Strunk, Kirsten. "Control of Personal Names" *Cataloging and Classification Quarterly.* 14.2(1991): 63-79.

Taylor, Arlene G. "Variations in Personal Name Access Points in OCLC Biographic Records" *Library Resources and Technical Services* 36(1992): 224-241.

Tillet, Barbara. 2001. "Authority Control on the Web" *Proceedings of the Bicentennial Conference on Bibliographic Control for the New Millennium* 10March 2005 <http://www.loc.gov/catdir/bibcontrol/tillett_paper.html>

Toivonen,Jarmo *et al* "Translating Cross-Lingual Spellings Variations Using Transformation Rules" *Information Processing and Management* 41(2005) 859-872.

Unicode Home Page. 2004.26 November 2004 <http://www.unicode.org>

Vassie, Roderic. "Improving Access in Bilingual, Biscript Catalogues Through Arabised Authority Control." *Online Information Review* 24(2000): 420-428.

Veerankutty Chelatayakkot and V. Jalaja "*Effectiveness of Name Searching in Web OPAC : From Authority Control to Access Control*" in Proceedings of the 3rd International CALIBER, Ahmadabad, Inflibnet, 2005 p.348-357

Warner, James W.and Elizabeth W. Brown. "Automated Name Authority Control" in *Proceedings of the First ACM/IEE-CS Joint Conference on Digital Libraries*: USA (2001) :21-22

Weintraub, Tamara S. "Personal Name Variations: Implications for Authority Control in Computerized Catalogues" *Library Resources and Technical Services* 35(1991): 217-228.

Yewang, Wang. "A look into Chinese Persons' Names in Bibliography Practice" *Cataloging and Classification Quarterly* 31.1 (2000): 51-81.

APPENDICES

Appendix I

Library of Congress
Romanization of Arabic

Isolated	From Right	Both Sides	From Left	Romanization	Name	Num Value
ا	ا	—	—	ʾ, ā *, **	*ʾalif*	1
ب	بـ	بـ	بـ	*b*	*bāʾ*	2
ت	تـ	تـ	تـ	*t*	*tāʾ*	400
ث	ثـ	ثـ	ثـ	*th*	*thāʾ*	500
ج	جـ	جـ	جـ	*j*	*jīm*	3
ح	حـ	حـ	حـ	*ḥ*	*ḥāʾ*	8
خ	خـ	خـ	خـ	*kh*	*khāʾ*	600
د	د	—	—	*d*	*dāl*	4
ذ	ذ	—	—	*dh*	*dhāl*	700
ر	ر	—	—	*r*	*rāʾ*	200
ز	ز	—	—	*z*	*zāy*	7
س	سـ	سـ	سـ	*s*	*sīn*	60
ش	شـ	شـ	شـ	*sh*	*shīn*	300
ص	صـ	صـ	صـ	*ṣ*	*ṣād*	90
ض	ضـ	ضـ	ضـ	*ḍ*	*ḍād*	800
ط	طـ	طـ	طـ	*ṭ*	*ṭāʾ*	9
ظ	ظـ	ظـ	ظـ	*ẓ*	*ẓāʾ*	900
ع	عـ	عـ	عـ	ʿ	*ʿayn*	70
غ	غـ	غـ	غـ	*gh*	*ghayn*	1000
ف	فـ	فـ	فـ	*f*	*fāʾ*	80
ق	قـ	قـ	قـ	*q*	*qāf*	100
ك	كـ	كـ	كـ	*k*	*kāf*	20
ل	لـ	لـ	لـ	*l*	*lām*	30
م	مـ	مـ	مـ	*m*	*mīm*	40
ن	نـ	نـ	نـ	*n*	*nūn*	50
ه	هـ، ة	هـ	هـ	*h, t* ***	*hāʾ*	5
و	و	—	—	*w,ū*	*wāw*	6
ى	ى	ـى	ـيـ	*y,ī*	*yāʾ*	10

Non-Arabic and Maghribī

Isolated	From Right	Both Sides	From Left	Romanization	Name
پ	پـ	ـپـ	ـپ	*p*	*pā'*
چ	چـ	ـچـ	ـچ	*ch*	*chīm*
ژ	ژ	—	—	*zh*	*zhāy*
ڿ	ڿـ	ـڿـ	ـڿ	*zh*	*zhīm*
گ	گـ	ـگـ	ـگ	*g*	*gāf*
ڤ	ڤـ	ـڤـ	ـڤ	*v*	*vā'*
ڢ	ڢـ	ـڢـ	ـڢ	*f*	*fā'*
ڧ	ڧـ	ـڧـ	ـڧ	*q*	*qāf*
ڤ	ڤـ	ـڤـ	ـڤ	*v*	*vā'*

Vowels and Diphthongs

ـَ	*a*	ـَا	*ā*	ـِي	*ī*
ـُ	*u*	ـَى ****	*á*	ـَوْ	*aw*
ـِ	*i*	ـُو	*ū*	ـَيْ	*ay*

* أ *hamzah* is omitted at the beginning of a word; elsewhere, it is written, eg., أَحمَد *Aḥmad*, but مُؤَسَّسَة *mu'assasah*, دَائِم *dā'im*, خَطِيء *khaṭī'*.

** The preposition لِ in combination with following article أَلْ is written *lil-*, e.g., لِلكِتَاب *lil-kitāb*, لِلشَّمس *lil-shams*. Note that the ل of the article is not assimilated to the "sun"-letters, e.g., أَلسَّفِير *al-safīr*.

*** ة is romanized *at* in a word in the construct state. To distinguish romanized ت in contact with another consonant from a roman digraph representing a single Arabic consonant, a single prime ′ is placed between the two distinct letters, e.g., *ad′ham* = أَدهَم, *akramat′hā* = أَكرَمَتهَا.

**** مُصطَفَى *Muṣṭafá*, but دَنَا *danā*.

Appendix II

SOURCE CODE

```
#include "main.h"
Bool prefix[MAXLENGTH];
Bool suffix[MAXLENGTH];
static int k;
static int nmbVowelsX2;
FilteredNameType filteredName[MAXLENGTH];

int main(const char * vbName)
{
   char userInputName[MAXLENGTH];
   InputNameType partsOfInputName[MAXLENGTH];
        FILE *fp;
        int i=0,j;
   int numSubNames;
        char seps[]   = " \t";
        char *token;
//      clrscr();

        fp=fopen("c:\\names.lst","w");
        if(fp==NULL)
        {
                //printf("Error in creating file\n");
                return (-1);
        }

   strcpy(userInputName,vbName);

        j=strlen(userInputName);

        printf("The length of %s name is %d\n",userInputName,j);

  token = strtok( userInputName, seps );
  while( token != NULL )
  {  strcpy(partsOfInputName[i].name,token);
    partsOfInputName[i].length=strlen(token);
    token = strtok( NULL, seps );
         i++;
  }
  numSubNames=i;
  printf("\nThe number of subnames is %d and they are\n",i);
  for(k=0;k<i;k++)
  {
          printf("%s length
%d\n",partsOfInputName[k].name,partsOfInputName[k].length);
```

```
}
i=0;
for(i=0;i<numSubNames;i++)
{
        if(strnicmp(partsOfInputName[i].name,"abdul",5) == 0)
        {
                partsOfInputName[i].prefix=TRUE;
                strcpy(partsOfInputName[i].name,partsOfInputName[i].name+5);
    partsOfInputName[i].length=strlen(partsOfInputName[i].name);
        }
        else if(strnicmp(partsOfInputName[i].name,"abdu",4) == 0)
        {
                partsOfInputName[i].prefix=TRUE;
                strcpy(partsOfInputName[i].name,partsOfInputName[i].name+4);
    partsOfInputName[i].length=strlen(partsOfInputName[i].name);

        }

        if(strrncmp(partsOfInputName[i].name,"dheen",5) == 0)
        {
                partsOfInputName[i].suffix=TRUE;
                partsOfInputName[i].name[partsOfInputName[i].length-5]='\0';
    partsOfInputName[i].length=strlen(partsOfInputName[i].name);
        }
        else if(strrncmp(partsOfInputName[i].name,"deen",4) == 0)
        {
                partsOfInputName[i].suffix=TRUE;
                partsOfInputName[i].name[partsOfInputName[i].length-4]='\0';
    partsOfInputName[i].length=strlen(partsOfInputName[i].name);
        }
        else if(strrncmp(partsOfInputName[i].name,"dhin",4) == 0)
        {
                partsOfInputName[i].suffix=TRUE;
                partsOfInputName[i].name[partsOfInputName[i].length-4]='\0';
    partsOfInputName[i].length=strlen(partsOfInputName[i].name);
        }
        else if(strrncmp(partsOfInputName[i].name,"ddin",4) == 0)
        {
                partsOfInputName[i].suffix=TRUE;
                partsOfInputName[i].name[partsOfInputName[i].length-4]='\0';
    partsOfInputName[i].length=strlen(partsOfInputName[i].name);
        }
        partsOfInputName[i].nmbVowels=findNmbVowels(partsOfInputName[i].name);
}
for(i=0;i<numSubNames;i++)
{
        if(partsOfInputName[i].length)
        {
```

```
        printf("partsOfInputName[%d]  is %s and its length %d
",i+1,partsOfInputName[i].name,partsOfInputName[i].length);
                printf("its vowelCount %d\n",partsOfInputName[i].nmbVowels);
        }
    else
                printf("THis has only prefix or suffix\n");
        if(partsOfInputName[i].prefix==TRUE)
                printf("its prefix abdul or abdu\n");
        else
                printf("its prefix is NULL\n");
        if(partsOfInputName[i].suffix==TRUE)
                printf("its suffix ddin\n");
        else
                printf("its suffix is NULL\n");
 }

 for(i=0;i<numSubNames;i++)
 {
        int indx;

        k=0;
        strcpy(filteredName[k].name,partsOfInputName[i].name);
   filteredName[k].length=partsOfInputName[i].length;
   filteredName[k].index=0;
        if (filteredName[k].length == 0)
                nmbVowelsX2=1;
        else
                nmbVowelsX2=2*findNmbVowels(filteredName[k].name) ;

        for(indx=0;indx<NMBFILTER;indx++)
      filteredName[k].filter[indx]=FALSE;
   filter(&filteredName[k]);

   if((partsOfInputName[i].prefix  == TRUE) && (partsOfInputName[i].suffix==
TRUE))
        {
                        for(indx=0;indx<=k && indx < nmbVowelsX2;indx++)
                        {
                fprintf(fp,"%s","abdul");
                        fprintf(fp,"%s",filteredName[indx].name);
                        fprintf(fp,"%s\n","ddin");
                        }
                        for(indx=0;indx<=k && indx < nmbVowelsX2;indx++)
                        {
                fprintf(fp,"%s","abdu");
                        fprintf(fp,"%s",filteredName[indx].name);
                        fprintf(fp,"%s\n","ddin");
                        }
```

```
        }
        else if((partsOfInputName[i].prefix== TRUE) &&
!(partsOfInputName[i].suffix== TRUE) )
        {
              for(indx=0;indx<=k && indx < nmbVowelsX2;indx++)
              {
                fprintf(fp,"%s","abdul ");
                fprintf(fp,"%s",filteredName[indx].name);
                fprintf(fp,"%s","\n");
              }
              for(indx=0;indx<=k && indx < nmbVowelsX2;indx++)
              {
                fprintf(fp,"%s","abdu ");
                fprintf(fp,"%s",filteredName[indx].name);
                fprintf(fp,"%s","\n");
              }
        }
        else if(!(partsOfInputName[i].prefix== TRUE) &&
(partsOfInputName[i].suffix== TRUE) )
        {
              for(indx=0;indx<=k && indx < nmbVowelsX2;indx++)
              {
                fprintf(fp,"%s",filteredName[indx].name);
                fprintf(fp,"%s\n","ddin");
              }
        }
        else
        {
              for(indx=0;indx<=k && indx < nmbVowelsX2;indx++)
              {
                fprintf(fp,"%s\n",filteredName[indx].name);
              }
        }
    fprintf(fp,"%d\n",777);
    printf("name %s index %d\n",filteredName[i].name,filteredName[i].index);
  }
  fclose(fp);
  /*printf("number of names %d \n",k+1);
  getch();*/

        return ROK;
}/*end of main  */


static int strrncmp(const char* s1, const char* s2, int num)
{
  int len1 = strlen(s1) - 1;
  int len2 = strlen(s2) - 1;
```

```
  if (len1 <0 || len1<0)
          return -1;
  for (; len1 >= 0 && len2 >= 0 && num > 0; len1--, len2--, num--) {
  const int d = (int)toupper(s1[len1]) - (int)toupper(s2[len2]);
  if (d != 0) return d;
  }
  return 0;
}

static int findNmbVowels(const char* s)
{
        int i=0,vowelCount=0;;
        for(i=0;s[i]!='\0';i++)
        {
                if( s[i] == 'a' || s[i] == 'A' ||
                        s[i] == 'e' || s[i] == 'E' ||
                        s[i] == 'i' || s[i] == 'I' ||
                        s[i] == 'o' || s[i] == 'O' ||
                        s[i] == 'u' || s[i] == 'U')
         vowelCount++;
        }
        return vowelCount;
}
static int filter(FilteredNameType *filterName)
{
        int i;
   printf("name %s\n",filterName->name);
        printf("index %d\n",filterName->index);
        printf("length %d\n",filterName->length);

        if(filterName->length == 0 )
                return ROK;

        filter10(&filteredName[0]);

        for(i=0;i<= k && k <=nmbVowelsX2;i++)
                filter34(&filteredName[i]);

        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filteredName[i].index=0;
        for(i=0;i<=k &&  k <= nmbVowelsX2;i++)
                filter19(&filteredName[i]);

        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filteredName[i].index=0;
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter40(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
```

```
                filter41(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter32(&filteredName[i]);

        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter29(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter27(&filteredName[i]);

        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filteredName[i].index=0;
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter15(&filteredName[i]);

        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filteredName[i].index=0;
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter17(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter20(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter24(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter25(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter30(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter33(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter38(&filteredName[i]);
        for(i=0;i<= k &&  k <= nmbVowelsX2;i++)
                filter39(&filteredName[i]);

        return ROK;
}
static int filter34(FilteredNameType *filterName)
{
        char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=34;
        filterName->length=strlen(filterName->name);
        for(;filterName->index<filterName->length;)
        {
                /*if(filterName->filter[filterIndex]==FALSE)
                {*/
                        charIndex=filterName->index;
                        ch=filterName->name[charIndex];
                        nextCh=filterName->name[charIndex+1];
```

```
                switch(ch)
                {


                        case 'K':
                        case 'k':
                                if(nextCh=='H' || nextCh=='h')
                                {

        makeCopyOfFilteredName(&filteredName[k+1],filterName);

        filteredName[k+1].name[charIndex]=(ch=='K')?'Q':'q';

        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;

        makeCopyOfFilteredName(&filteredName[k+2],filterName);

        strcpy(filteredName[k+2].name+charIndex+1,filterName->name+charIndex+2);
                                        filteredName[k+2].index+=1;
                                        filteredName[k+2].filter[filterIndex]=TRUE;

                                        filterName->index+=2;
                                        filterName->filter[filterIndex]=TRUE;
                                        k+=2;
                                }
                                else
#if 0 /* Modification #6 */
                                {

        makeCopyOfFilteredName(&filteredName[k+1],filterName);

        filteredName[k+1].name[charIndex]=(ch=='K')?'Q':'q';
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;

        makeCopyOfFilteredName(&filteredName[k+2],filterName);

        filteredName[k+2].name[charIndex+1]=(ch=='K')?'H':'h';

        strcpy(filteredName[k+2].name+charIndex+2,filterName->name+charIndex+1);
                                        filteredName[k+2].index+=2;
                                        filteredName[k+2].filter[filterIndex]=TRUE;
                                        filterName->index+=1;
                                        filterName->filter[filterIndex]=TRUE;
                                        k+=2;
                                }
```

```
#endif
                                        filterName->index+=1;
                                        break;

                        case 'Q':
                        case 'q':

        makeCopyOfFilteredName(&filteredName[k+1],filterName);

        filteredName[k+1].name[charIndex]=(ch=='Q')?'K':'k';
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;

        makeCopyOfFilteredName(&filteredName[k+2],filterName);
                filteredName[k+2].name[charIndex]=(ch=='Q')?'K':'k';
                filteredName[k+2].name[charIndex+1]=(ch=='Q')?'H':'h';

        strcpy(filteredName[k+2].name+charIndex+2,filterName->name+charIndex+1);
                                        filteredName[k+2].index+=2;
                                        filteredName[k+2].filter[filterIndex]=TRUE;
                                        filterName->index+=1;;
                                        filterName->filter[filterIndex]=TRUE;
                                        k+=2;
                                        break;
                                case 'S':
                                case 's':
                                        if(nextCh=='H' || nextCh=='h')
                                {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                                filteredName[k+1].index+=1;
                                                filteredName[k+1].filter[filterIndex]=TRUE;
                                                filterName->index+=2;
                                                filterName->filter[filterIndex]=TRUE;
                                                k+=1;
                                        }
                        default:
                                filterName->index+=1;
                                break;
                        }
                /*}*/
        }
  return ROK;
}
static int  makeCopyOfFilteredName(FilteredNameType *dst,FilteredNameType *src)
{
  memcpy((void *)dst,(void *)src,sizeof(FilteredNameType));
        return ROK;
```

```
}
/*if the first letter is I or E or O or U or S or Z */
static int filter10(FilteredNameType *filterName)
{
        char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=10;

        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=0;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
                switch(ch)
                {
#if 0   /* modification #2 by */
                    case 'I':
                        case 'i':
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                filteredName[k+1].name[charIndex]=(ch=='I')?'E':'e';
                                filteredName[k+1].index+=1;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index+=1;
                                filterName->filter[filterIndex]=TRUE;
                                k+=1;
                                break;
#endif

                        case 'E':
                        case 'e':
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                filteredName[k+1].name[charIndex]=(ch=='E')?'I':'i';
                                filteredName[k+1].index+=1;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index+=1;
                                filterName->filter[filterIndex]=TRUE;
                                k+=1;
                                break;

                        case 'O':
                        case 'o':

        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                filteredName[k+1].name[charIndex]=(ch=='O')?'U':'u';
                                filteredName[k+1].index+=1;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index+=1;
                                filterName->filter[filterIndex]=TRUE;
```

```
                        k+=1;
                        break;


                case 'U':
                case 'u':
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                        filteredName[k+1].name[charIndex]=(ch=='U')?'O':'o';
                        filteredName[k+1].index+=1;
                        filteredName[k+1].filter[filterIndex]=TRUE;
                        filterName->index+=1;
                        filterName->filter[filterIndex]=TRUE;
                        k+=1;
                        break;
#if 0 /* modification  #3 */
                case 'S':
                case 's':
                        if(nextCh!='H' && nextCh!='h')
                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                filteredName[k+1].name[charIndex]=(ch=='S')?'Z':'z';
                                filteredName[k+1].index+=1;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index+=1;
                                filterName->filter[filterIndex]=TRUE;
                                k+=1;
                        }
                        break;
#endif

                case 'Z':
                case 'z':
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                        filteredName[k+1].name[charIndex]=(ch=='Z')?'S':'s';
                        filteredName[k+1].index+=1;
                        filteredName[k+1].filter[filterIndex]=TRUE;
                        filterName->index+=1;
                        filterName->filter[filterIndex]=TRUE;
                        k+=1;
                        break;
                case 'A':
                case 'a':
                        filterName->index+=1;
                        filterName->filter[filterIndex]=TRUE;
                        break;
                /*}*/
        }
  return ROK;
```

```
}


static int filter15(FilteredNameType *filterName)
{
        char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=15;
        for(;filterName->index<filterName->length;)
        {
                /*if(filterName->filter[filterIndex]==FALSE)
                  {*/
                charIndex=filterName->index;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
                switch(ch)
                {
                        case 'T':
                        case 't':
                                if(nextCh=='H' || nextCh=='h')
                                {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        filterName->index+=2;
                                        filterName->filter[filterIndex]=TRUE;
                                        if(isNotRepeat(&filteredName[k+1]))
                                                k++;
                                }
                                else
                                {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex+1]=(ch=='T')?'H':'h';
        strcpy(filteredName[k+1].name+charIndex+2,filterName->name+charIndex+1);
                                        filteredName[k+1].index+=2;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        filterName->index+=1;
                                        filterName->filter[filterIndex]=TRUE;
                                        if(isNotRepeat(&filteredName[k+1]))
                                                k++;
                                }
                                break;
                        case 'V':
                        case 'v':
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex]=(ch=='V')?'W':'w';
```

```
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        filterName->index+=1;
                                        filterName->filter[filterIndex]=TRUE;
                                        if(isNotRepeat(&filteredName[k+1]))
                                                k++;
                                        break;
                        case 'W':
                        case 'w':
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex]=(ch=='W')?'V':'v';
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        filterName->index+=1;
                                        filterName->filter[filterIndex]=TRUE;
                                        if(isNotRepeat(&filteredName[k+1]))
                                                k++;
                                        break;
                        default:
                                filterName->index+=1;
                                break;
                }
        }
   return ROK;
}
static int filter29(FilteredNameType *filterName)
{
        char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=29;
        for(;filterName->index<filterName->length;)
        {
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=filterName->index;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
                switch(ch)
                {
                        case 'O':
                        case 'o':
                                if(nextCh=='o' || nextCh=='O')
                                {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex]=(ch=='O')?'U':'u';
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
```

```
                                        if(isNotRepeat(&filteredName[k+1])      &&
                                          isNotTwoIiOrUu(&filteredName[k+1],'u') &&
                                          isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                                k++;
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex+1]=(ch=='O')?'U':'u';
                                        filteredName[k+1].index+=2;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        if(isNotRepeat(&filteredName[k+1])      &&
                                          isNotTwoIiOrUu(&filteredName[k+1],'u') &&
                                          isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                                k++;
                                        filterName->index+=2;
                                        filterName->filter[filterIndex]=TRUE;
                                }
                                else
                                {
                                        if(nextCh=='u' || nextCh=='U')
                                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex]=(ch=='O')?'U':'u';
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                                filteredName[k+1].index+=1;
                                                filteredName[k+1].filter[filterIndex]=TRUE;
                                                filterName->index+=2;
                                                filterName->filter[filterIndex]=TRUE;
                                                if(isNotRepeat(&filteredName[k+1])      &&
                                                  isNotTwoIiOrUu(&filteredName[k+1],'u')
&&
                                                  isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                                        k++;
                                        }
                                        else
                                                filterName->index+=1;
                                }
                                break;
                        default:
                                filterName->index+=1;
                                break;
                }
        }
  return ROK;
}

static int filter20(FilteredNameType *filterName)
{
        char ch,nextCh;
  int filterIndex,charIndex;
```

```
filterIndex=20;
        for(;filterName->index<filterName->length;)
        {

/*      if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=filterName->index;
                if(charIndex==0) charIndex=1;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
                switch(ch)
                {
                        case 'I':
                        case 'i':
                                if(nextCh=='A' || nextCh=='a')
                                {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex]=(ch=='I')?'Y':'y';
                                        filteredName[k+1].index+=2;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        filterName->index+=2;
                                        filterName->filter[filterIndex]=TRUE;
                                        if(isNotRepeat(&filteredName[k+1]))
                                                k++;
                                }
                                else
                                {
                                        if(nextCh=='E' || nextCh=='e')
                                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex]=(ch=='I')?'E':'e';
                                                filteredName[k+1].index+=2;
                                                filteredName[k+1].filter[filterIndex]=TRUE;
                                                if(isNotRepeat(&filteredName[k+1]))
                                                        k++;
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                                filteredName[k+1].index+=1;
                                                filteredName[k+1].filter[filterIndex]=TRUE;

                                                filterName->index+=2;
                                                filterName->filter[filterIndex]=TRUE;
                                                if(isNotRepeat(&filteredName[k+1]))
                                                        k++;
                                        }
                                        else
                                                filterName->index+=1;
                                }
```

```
                    break;


                case 'Y':
                case 'y':
                        if(nextCh=='A' || nextCh=='a')
                        {
makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                filteredName[k+1].name[charIndex]=(ch=='Y')?'I':'i';
                                filteredName[k+1].index+=2;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index+=2;
                                filterName->filter[filterIndex]=TRUE;
                                if(isNotRepeat(&filteredName[k+1])      &&
                                  isNotTwoIiOrUu(&filteredName[k+1],'u') &&
                                  isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                        k++;
                        }
                        else
                                 filterName->index+=1;
                        break;
                case 'A':
                case 'a':
                        if(nextCh=='I' || nextCh=='i')
                        {
makeCopyOfFilteredName(&filteredName[k+1],filterName);
filteredName[k+1].name[charIndex+1]=(ch=='A')?'Y':'y';
                                filteredName[k+1].index+=2;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index+=2;
                                filterName->filter[filterIndex]=TRUE;
                                if(isNotRepeat(&filteredName[k+1]))
                                        k++;
                        }
                        else
                        {
                                if(nextCh=='Y' || nextCh=='y')
                                {
makeCopyOfFilteredName(&filteredName[k+1],filterName);
filteredName[k+1].name[charIndex+1]=(ch=='A')?'I':'i';
                                        filteredName[k+1].index+=2;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        filterName->index+=2;
                                        filterName->filter[filterIndex]=TRUE;

                                        if(isNotRepeat(&filteredName[k+1])      &&
```

```
                                        isNotTwoIiOrUu(&filteredName[k+1],'u')
&&
                                        isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                                k++;
                                }
                                else
                                        filterName->index+=1;
                        }
                        break;
                case 'E':
                case 'e':
                        if(nextCh=='I' || nextCh=='i')
                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex+1]=(ch=='E')?'Y':'y';
                                filteredName[k+1].index+=2;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index+=2;
                                filterName->filter[filterIndex]=TRUE;
                                if(isNotRepeat(&filteredName[k+1]))
                                        k++;;
                        }
                        else
                        {
                                if(nextCh=='Y' || nextCh=='y')
                                {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex+1]=(ch=='E')?'I':'i';
                                        filteredName[k+1].index+=2;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        filterName->index+=2;
                                        filterName->filter[filterIndex]=TRUE;
                                        if(isNotRepeat(&filteredName[k+1])      &&
                                          isNotTwoIiOrUu(&filteredName[k+1],'u')
&&
                                          isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                                k++;
                                }
                                else
                                {
                                        if(nextCh=='E' || nextCh=='e')
                                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex]=(ch=='E')?'I':'i';
        filteredName[k+1].index+=2;
        filteredName[k+1].filter[filterIndex]=TRUE;
                                                if(isNotRepeat(&filteredName[k+1])
&&
```

```
isNotTwoIiOrUu(&filteredName[k+1],'u') &&
isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                                        k++;

        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex]=(ch=='E')?'I':'i';
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                                        filteredName[k+1].index+=1;
        filteredName[k+1].filter[filterIndex]=TRUE;
                                                        filterName->index+=2;
                                                        filterName-
>filter[filterIndex]=TRUE;
                                                        if(isNotRepeat(&filteredName[k+1])
&&
isNotTwoIiOrUu(&filteredName[k+1],'u') &&
isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                                        k++;
                                                }
                                                else
                                                        filterName->index+=1;
                                        }
                                }
                                break;
                        default:
                                filterName->index+=1;
                                break;
                }
        }
  return ROK;
}

static int filter25(FilteredNameType *filterName)
{
        char ch,nextCh;
  int filterIndex,charIndex;
  filterIndex=25;
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=filterName->index;
                ch=filterName->name[1];
                nextCh=filterName->name[filterName->length-2];

                if (( ch =='a' || ch=='A' ) &&   (nextCh=='i' || nextCh=='I'))
                {
                makeCopyOfFilteredName(&filteredName[k+1],filterName);
                filteredName[k+1].name[filteredName[k+1].length-2]=(ch=='A')?'E':'e';
                filteredName[k+1].filter[filterIndex]=TRUE;
```

```
                if(isNotRepeat(&filteredName[k+1]))
                        k++;
                }


        /*}*/
   return ROK;
}
static int nextVowel(const char* s,int cnt)
{
        int i=0,vowelCount=0;;
        for(i=0;s[i]!='\0';i++)
        {
                if( s[i] == 'a' || s[i] == 'A' ||
                        s[i] == 'e' || s[i] == 'E' ||
                        s[i] == 'i' || s[i] == 'I' ||
                        s[i] == 'o' || s[i] == 'O' ||
                        s[i] == 'u' || s[i] == 'U' )
         vowelCount++;
         if(vowelCount==cnt)
           return i;
        }
        return -1;
}
static int isVowel(char s)
{
                if( s == 'a' || s == 'A' ||
                   s == 'e' || s == 'E' ||
                   s == 'i' || s == 'I' ||
                   s == 'o' || s == 'O' ||
                   s == 'u' || s == 'U' )
         return 1;
         return 0;
}
static int filter16(FilteredNameType *filterName)
{
   char ch,nextCh;
   int filterIndex,charIndex=0;
   filterIndex=16;
        ch=nextCh=filterName->name[filterName->length-1];
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                if(isVowel( ch))
                printf("no change\n");
                filterName->filter[filterIndex]=TRUE;
        /*}*/
        return ROK;
}
```

```
static int filter17(FilteredNameType *filterName)
{
   char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=17;
   charIndex=0;
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=0;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
                if( !isVowel(ch) && !isVowel(nextCh) )
                {
                 /* no change to first vowel*/
                 charIndex=nextVowel(filterName->name,1);
                 if(charIndex != -1)
                 {  ch=filterName->name[charIndex];
                   while(filterName->name[charIndex++]!=ch) ;
                        filterName->index=charIndex;
                 }
                }
                filterName->filter[filterIndex]=TRUE;
        /*}*/
        return ROK;
}

static int filter19(FilteredNameType *filterName)
{
        char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=19;
        for(;filterName->index<filterName->length;)
        {
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=filterName->index;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
                switch(ch)
                {
                        case 'K':
                        case 'k':
                                if(nextCh=='K' || nextCh=='k')
                                {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
```

```
                                filterName->index+=1;
                                k+=1;
                        }

                        filterName->index+=1;
                        filterName->filter[filterIndex]=TRUE;
                        break;
                case 'S':
                case 's':
                        if(nextCh=='S' || nextCh=='s')
                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                filteredName[k+1].index+=1;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index+=1;
                                k+=1;
                        }
                        filterName->index+=1;
                        filterName->filter[filterIndex]=TRUE;
                        break;
                case 'M':
                case 'm':
                        if(nextCh=='M' || nextCh=='m')
                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                filteredName[k+1].index+=1;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index+=1;
                                k+=1;
                        }
                        filterName->index+=1;
                        filterName->filter[filterIndex]=TRUE;
                        break;

/* new condition #51 */
                case 'T':
                case 't':
                        if(nextCh=='T' || nextCh=='t')
                        {
                                strcpy(filterName->name+charIndex+1,filterName-
>name+charIndex+2);
                        }
                        filterName->index+=1;
                        filterName->filter[filterIndex]=TRUE;
                        break;
                case 'W':
```

```
                        case 'w':
                                if(nextCh=='W' || nextCh=='w')
                                {

                strcpy(filterName->name+charIndex+1,filterName->name+charIndex+2);
                                }
                                filterName->index+=1;
                                filterName->filter[filterIndex]=TRUE;
                                break;
                        case 'V':
                        case 'v':
                                if(nextCh=='V' || nextCh=='v')
                                {
                strcpy(filterName->name+charIndex+1,filterName->name+charIndex+2);
                                }
                                filterName->index+=1;
                                filterName->filter[filterIndex]=TRUE;
                                break;
                        case 'B':
                        case 'b':
                                if(nextCh=='B' || nextCh=='b')
                                {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        strcpy(filteredName[k+1].name+charIndex+1,filterName->name+charIndex+2);
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        filterName->index+=1;
                                        k+=1;
                                }
                                filterName->index+=1;
                                filterName->filter[filterIndex]=TRUE;
                                break;
                        default:
                                  filterName->index+=1;
                                  break;
                }
        }
        return ROK;
}

static int filter24(FilteredNameType *filterName)
{
        char ch,nextCh;
        int filterIndex,charIndex,prevCh;
        filterIndex=24;
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=0;
```

```
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
        charIndex =nextVowel(filterName->name,findNmbVowels(filterName->name));
                if(charIndex!=-1)
                {
                        ch=filterName->name[charIndex];
                        prevCh=filterName->name[charIndex-1];
                        if( (ch=='i' || ch=='I')  && (prevCh!='a' && prevCh!='A' &&
prevCh!='e' && prevCh!='E') )
                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                filteredName[k+1].name[charIndex+1]=(ch=='I')?'E':'e';
        strcpy(filteredName[k+1].name+charIndex+2,filterName->name+charIndex+1);
                                filteredName[k+1].index+=2;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                if(isNotRepeat(&filteredName[k+1]))
                                        k++;
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                filteredName[k+1].name[charIndex]=(ch=='I')?'E':'e';
                                filteredName[k+1].name[charIndex+1]=(ch=='I')?'E':'e';
        strcpy(filteredName[k+1].name+charIndex+2,filterName->name+charIndex+1);
                                filteredName[k+1].index+=2;
                                filteredName[k+1].filter[filterIndex]=TRUE;
                                if(isNotRepeat(&filteredName[k+1]))
                                        k++;
                        }
                }
                filterName->filter[filterIndex]=TRUE;
        /*}*/
  return ROK;
}

static int filter27(FilteredNameType *filterName)
{
        char ch,nextCh,prevCh;
  int filterIndex,charIndex;
  filterIndex=27;
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
          filterName->length=strlen(filterName->name);
                charIndex=0;
                ch=filterName->name[charIndex];
                charIndex=filterName->length-2;
                nextCh=filterName->name[charIndex];
                prevCh=filterName->name[charIndex-1];
                if( (nextCh=='u' || nextCh=='U') &&
                        (ch !='y' &&  ch !='Y')     &&
                        (prevCh !='O' &&  prevCh !='o'))
```

```
                {


makeCopyOfFilteredName(&filteredName[k+1],filterName);
                        filteredName[k+1].name[charIndex]=(ch=='U')?'O':'o';
                        filteredName[k+1].name[charIndex+1]=(ch=='U')?'O':'o';
strcpy(filteredName[k+1].name+charIndex+2,filterName->name+charIndex+1);
                        filteredName[k+1].index+=2;
                        filteredName[k+1].filter[filterIndex]=TRUE;
                        if(isNotRepeat(&filteredName[k+1]))
                                k++;
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                        filteredName[k+1].name[charIndex]=(ch=='U')?'O':'o';
                        filteredName[k+1].name[charIndex+1]=(ch=='U')?'U':'u';
        strcpy(filteredName[k+1].name+charIndex+2,filterName->name+charIndex+1);
                        filteredName[k+1].index+=2;
                        filteredName[k+1].filter[filterIndex]=TRUE;
                        if(isNotRepeat(&filteredName[k+1])      &&
                          isNotTwoIiOrUu(&filteredName[k+1],'u') &&
                          isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                k++;
                }
                filterName->filter[filterIndex]=TRUE;
        /*}*/
  return ROK;
}

static int filter30(FilteredNameType *filterName)
{
  char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=30;
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=1;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
     if( (ch=='i' || ch == 'I') && (nextCh != 'a' && nextCh!= 'A'))
                {

        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                        filteredName[k+1].name[charIndex]=(ch=='I')?'Y':'y';
                        filteredName[k+1].index+=1;
                filteredName[k+1].filter[filterIndex]=TRUE;
                        if(isNotRepeat(&filteredName[k+1]))
                                k++;
                }
```

```
                else
                {


if( (ch=='y' || ch == 'Y') && (nextCh != 'a' && nextCh!= 'A'))
                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                filteredName[k+1].name[charIndex]=(ch=='Y')?'I':'i';
                                filteredName[k+1].index+=1;
                filteredName[k+1].filter[filterIndex]=TRUE;
                                if(isNotRepeat(&filteredName[k+1])      &&
                                  isNotTwoIiOrUu(&filteredName[k+1],'u') &&
                                  isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                        k++;
                        }
                }
                filterName->filter[filterIndex]=TRUE;
        /*}*/
   return ROK;
}

static int filter32(FilteredNameType *filterName)
{
        char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=32;
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=0;
                ch=filterName->name[charIndex];
          if( ch=='O' || ch =='o' || ch == 'U' || ch =='u')
                {
                        ch=filterName->name[charIndex+1];
                        nextCh=filterName->name[charIndex+2];
                        if( (ch=='t' || ch =='T') &&
          (nextCh=='h' || nextCh =='H') )
                        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                filteredName[k+1].name[charIndex+1]=(ch=='T')?'S':'s';
        strcpy(filteredName[k+1].name+charIndex+2,filterName->name+charIndex+3);
                                filteredName[k+1].index=2;
                filteredName[k+1].filter[filterIndex]=TRUE;
                                filterName->index=3;
                                filterName->filter[filterIndex]=TRUE;
                                if(isNotRepeat(&filteredName[k+1]))
                                        k++;
                        }
                }
```

```
        /*}*/
   return ROK;
}

static int filter33(FilteredNameType *filterName)
{
        char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=33;
/*      if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=1;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
                if(!isVowel(ch) && !isVowel(nextCh))
                {
       if(isVowel(filterName->name[charIndex-1]))
         charIndex=nextVowel(filterName->name,2);
                  else
         charIndex=nextVowel(filterName->name,1);
                  if(charIndex != -1)
                  {
                          ch=filterName->name[charIndex];
                          if(ch=='E' || ch =='e')
                          {
makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                  filteredName[k+1].name[charIndex]=(ch=='E')?'A':'a';
                                  filteredName[k+1].index+=1;
                                  filteredName[k+1].filter[filterIndex]=TRUE;
                                 if(isNotRepeat(&filteredName[k+1]))
                                         k++;
                          }
                  }
                }
                filterName->filter[filterIndex]=TRUE;
/*      }*/
   return ROK;
}
static int filter38(FilteredNameType *filterName)
{
        char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=38;
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=1;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
```

```
            if( (ch == 'E' || ch =='e')        &&
               (nextCh !='i' && nextCh != 'I') &&
               (nextCh !='y' && nextCh != 'Y') &&
                     (nextCh !='e' && nextCh != 'E')        )
            {

      makeCopyOfFilteredName(&filteredName[k+1],filterName);
                           filteredName[k+1].name[charIndex]=(ch=='E')?'A':'a';
                           filteredName[k+1].index+=1;
            filteredName[k+1].filter[filterIndex]=TRUE;
                           if(isNotRepeat(&filteredName[k+1]))
                                 k++;
            }
            filterName->filter[filterIndex]=TRUE;
      /*}*/
   return ROK;
}

static int filter39(FilteredNameType *filterName)
{
   char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=39;

/*       if(filterName->filter[filterIndex]==FALSE)
         {*/
            filterName->length=strlen(filterName->name);

      charIndex =nextVowel(filterName->name,findNmbVowels(filterName->name));
            if(charIndex!=-1)
            {
                  ch=filterName->name[charIndex];
                  nextCh=filterName->name[charIndex];
                  switch(ch)
                  {
                     case 'A':
                     case 'a':
                                    if(charIndex != (filterName->length-1))
                                    {

      makeCopyOfFilteredName(&filteredName[k+1],filterName);
      filteredName[k+1].name[charIndex]=(ch=='A')?'E':'e';
                                          filteredName[k+1].index+=1;
                                          filteredName[k+1].filter[filterIndex]=TRUE;
                                          if(isNotRepeat(&filteredName[k+1]))
                                                k++;
                                    }
                                    break;
```

```
                                case 'E':
                                case 'e':
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex]=(ch=='E')?'A':'a';
                                        filteredName[k+1].index+=1;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        if(isNotRepeat(&filteredName[k+1]))
                                                k++;
                                break;
                        }
                }
                filterName->filter[filterIndex]=TRUE;
        /*}*/
   return ROK;
}
static int filter40(FilteredNameType *filterName)
{
        char ch,nextCh;
   int filterIndex,charIndex;
   filterIndex=40;
        /*if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=0;
     ch=filterName->name[charIndex];
                if( ch=='Y' || ch =='y')
                {
                        charIndex =nextVowel(filterName->name,1);
                        if(charIndex!=-1)
                        {
                                ch=filterName->name[charIndex];
                                nextCh=filterName->name[charIndex+1];
                          if((ch =='O' || ch =='o') && (nextCh != 'O' && nextCh!='o') )
                                {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
        filteredName[k+1].name[charIndex+1]=(ch=='O')?'U':'u';
        strcpy(filteredName[k+1].name+charIndex+2,filterName->name+charIndex+1);
                                        filteredName[k+1].index=charIndex+2;
                                        filteredName[k+1].filter[filterIndex]=TRUE;
                                        filterName->index=charIndex;
                                        filterName->filter[filterIndex]=TRUE;
                                        if(isNotRepeat(&filteredName[k+1])      &&
                                          isNotTwoIiOrUu(&filteredName[k+1],'u') &&
                                          isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                                k++;
                                }
                        }
                }
        /*}*/
```

```
  return ROK;
}
static int filter41(FilteredNameType *filterName)
{
        char ch,nextCh;
  int filterIndex,charIndex;
  filterIndex=41;
/*      if(filterName->filter[filterIndex]==FALSE)
        {*/
                charIndex=1;
                ch=filterName->name[charIndex];
                nextCh=filterName->name[charIndex+1];
                if( (ch == 'O' || ch =='o') && (nextCh!= 'O' && nextCh != 'o'))
        {
        makeCopyOfFilteredName(&filteredName[k+1],filterName);
                                filteredName[k+1].name[charIndex]=(ch=='O')?'U':'u';
                                filteredName[k+1].index+=1;
                filteredName[k+1].filter[filterIndex]=TRUE;
                filterName->index=2;
                filterName->filter[filterIndex]=TRUE;
                                if(isNotRepeat(&filteredName[k+1])      &&
                                 isNotTwoIiOrUu(&filteredName[k+1],'u') &&
                                  isNotTwoIiOrUu(&filteredName[k+1],'i') )
                                        k++;
                }
        /*}*/
  return ROK;
}
int isNotRepeat(FilteredNameType *filterName)
{
        int i;
        for(i=0;i<=k;i++)
        {
                if(!strcmp(filteredName[i].name,filterName->name))
                        return 0;
        }
        return  1;
}
int isNotTwoIiOrUu(FilteredNameType *filterName,char ch)
{
        int i;
        for (i=0;filterName->name[i]!='\0';i++)
if(tolower(filterName->name[i])==tolower(ch) && tolower(filterName-
>name[i+1])==tolower(ch))
                        return 0;
        }
        return 1;
}
```

APPENDIX III

Software

The accompanying CD-ROM contains two software:

1. Arabic Name Searcher.

This program is used to create variant spelling of Romanised Arabic Names. This program can be run as follows.

a) Double click 'Arabic Name Searcher' Folder.

b) Double click 'VB' folder.

c) Double click 'User Interface' icon

The source code of this program is available in 'VC' folder named as 'source code'

2. Web Interface

This program shows how we can use 'Arabic Name Searcher' can be used as interface in Database name searching and Free Text name searching.

Requirements: Windows 2000/ Windows NT with Microsoft Internet Information Server (IIS)

Installation: (a) Copy file 'searchnames.asp' to the folder 'C:\Inetpub\wwwroot'

(b) Copy 'myDll.dll' to 'C:\winNT\Systems 32

(c) In DOS prompt, go to SearchNames .DLL and enter 'regsvr32 SearchNames.dll'.

(d) Open the searchname.asp in any Browser and open 'http://localhost/searchname.asp'