



Vrije Universiteit Brussel

FACULTY OF ENGINEERING

# Extension with Digital Library Technology of UNESCO's J-ISIS Database Software

---

**By: Helen Hagos Berhe**

---

**Promoter: Prof. dr. Eddy Vandijck**

**Co-promoter: Prof. dr. Egbert de Smet**

Thesis submitted in partial fulfilment of the requirements for the Master of  
Science in Applied Sciences and Engineering: Applied Computer Science

**Academic year 2011-2012**



## **Acknowledgement**

First of all I would like to thank my promoter Prof. dr. Eddy Vandijck for his support, encouragement and patience and my co-promoter Prof. dr. Egbert de Smet for his guidance, precious support and comments. I would like also to extend my gratitude to Mr. Jean-Claude Dauphin (UNESCO, Paris) who assisted me in working with J-ISIS, Prof. dr. Jacques Tiberghien for his concern, support and encouragements, and Mrs. Irene Raadschelders for her support and encouragements for finishing my study program. Last but not least, I would like to thank my family, especially my father, and my friends who supported and motivated me for the completion of my thesis.

## Table of Contents

Acknowledgement.....	i
Acronyms.....	v
List of Figures.....	vii
List of Tables.....	vii
Abstract.....	viii
Chapter One: Introduction and Problem Statement.....	- 1 -
1.1. Introduction.....	- 1 -
1.2. The Digital Library Concept .....	- 1 -
1.3. Web Information Systems .....	- 4 -
1.4. Some Technological Requirements for Digital Libraries .....	- 6 -
1.5. ISIS and ABCD for Digital Library Applications.....	- 6 -
1.6. Objective of the Study.....	- 8 -
1.7. Significance of the Study .....	- 9 -
1.8. Methodology .....	- 9 -
Chapter Two: Discussion of Some Technical Requirements of Digital Libraries .....	- 10 -
2.1. WWW-based Retrieval .....	- 10 -
2.2. Diversity of Document Formats .....	- 11 -
2.3. Diversity of Metadata Formats.....	- 12 -
2.4. Full-text Search and Ranking .....	- 13 -
2.4.1. Full-text Search .....	- 13 -
2.4.2. Ranking.....	- 14 -
2.5. Unicode .....	- 14 -
2.6. Open Archives Initiative-Protocol for Metadata Harvesting (OAI-PMH) .....	- 16 -
2.7. Interactivity.....	- 20 -
Chapter Three: History and Technical Concepts of the Classical ISIS Technology and ABCD Software.....	- 21 -
3.1. Introduction on ISIS.....	- 21 -

3.2.	Brief History of the ISIS Technology.....	- 21 -
3.3.	Technical Concepts of ISIS as a Semi-Structured NoSQL Database.....	- 28 -
3.3.1.	The ISIS Master File (MST) .....	- 30 -
3.3.2.	The ISIS Cross-Reference File (XRF).....	- 32 -
3.3.3.	The ISIS Inverted File (IF) .....	- 32 -
3.3.4.	The ISIS Query Language .....	- 32 -
3.3.5.	The ISIS Formatting Language .....	- 33 -
Chapter Four: Technical Concepts of the New Generation ISIS Software (J-ISIS) .....		- 36 -
4.1.	Introduction.....	- 36 -
4.2.	Java Integrated Set for Information Services (J-ISIS): The Client-Server Java Implementation of ISIS - 36 -	
4.3.	Web-JISIS .....	- 37 -
4.4.	NoSQL Database Technologies .....	- 39 -
4.5.	Berkeley DB.....	- 43 -
4.5.1.	Berkeley DB Product Family Overview .....	- 44 -
4.5.2.	Comparison among the Berkeley DB Family Products .....	- 44 -
4.5.3.	How Berkeley DB is Used.....	- 46 -
4.6.	Lucene .....	- 49 -
4.7.	Comparison between ISIS and J-ISIS .....	- 51 -
Chapter Five: Implementing and Testing Digital Library Technology into J-ISIS .....		- 54 -
5.1.	Testing Available Digital Library Features in J-ISIS.....	- 54 -
5.1.1.	WWW Based Retrieval: Web-JISIS.....	- 54 -
5.1.2.	Diversity of Metadata Formats.....	- 56 -
5.1.3.	Full-text Indexing .....	- 57 -
5.1.4.	Unicode.....	- 71 -
5.2.	Implementation of the Digital Library Feature .....	- 74 -
5.2.1.	Presentation of the Newly Added Digital Library Feature .....	- 74 -
5.2.2.	Technical Discussion of the Newly Implemented Digital Library Feature.....	- 85 -

5.2.3. Testing J-ISIS Performance .....	- 95 -
5.3. The OAI-PMH .....	- 98 -
5.4. Interactivity.....	- 99 -
Chapter Six: Conclusions and Recommendations.....	- 100 -
References.....	- 103 -

## Acronyms

ABCD	Automation of liBRaries and Centers for Documentation
ACID	Atomicity, Consistency, Isolation, Durability
AJAX	Asynchronous JavaScript and XML
ANSI	American National Standard Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BDB	Berkeley Database
Bsearch	Binary search
CCF	Common Communication Format
CDS	Computerized Documentation Service
CDS/ISIS	Computerized Documentation Service/Integrated Set for Information Services
CEPAL	(Bibliographic standard of the) Comision Economica para America Latina
CGI	Common Gateway Interface
CISIS	ISIS programmed in C (Brazil)
CMS	Content Management System
DBA	Documentazione, le Biblioteche e gli Archivi: (Italian association for documentalists and librarians)
Dbm	Database Manager
DC	Dublin Core
DL	Digital Library
DLL	Dynamically Linked Library
DOS	Disk Operating Systems
FOSS	Free and Open Source Software
FST	Field Selection Table
FTS	Full-text Search
GSDL	GreenStone Digital Library
Gdbm	GNU database manager
GMOD	Global Modifications in ISIS databases
GNU	GNU's Not Unix
HP	Hewlett-Packard
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machine
IDRC	International Development Research Centre
IF	Inverted File
ILO	International Labour Organization
IM	Instant Messaging
IP	Internet Protocol
IR	Information Retrieval
ISIS	Integrated Set for Information Services
ISO	International Organization for Standardization
ISRS	Information Storage and Retrieval Systems
ISISDLL	Integrated Set for Information Services Dynamically linked Library
ISIS-NBP	ISIS Network Based Platform.

JAVASIS	JAVA Integrated Set of Information Services
J-ISIS	Java- Integrated Set for Information Services
JSON	Java Script Object Notation
JSP	Java Server pages
MINISIS	ISIS for Minicomputers (Canada)
MARC	MAchine-Readable Cataloging
MARC 21	MAchine-Readable Cataloging
MFN	Master File Number
MST	ISIS Master File
MVC	Model-View-Controller
Mx	Master/Cross-reference files utility for ISIS
NCSA	National Center for Supercomputing Applications
Ndbm	New database manager
NoSQL	Not only SQL
NIO	New Input Output
OAI-PMH	Open Archives Initiative-Protocol for Metadata Harvesting
OPAC	Online Public Access Catalogue
OpenIsis	Open source ISIS (Germany)
PDF	Portable Document Format
PFT	Print Format Table
PHP	Hypertext Pre-processor
RIA	Rich Internet Application
RSS	Really Simple Syndication
RTF	Rich Text Format
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TXT/HTML	Text/ HyperText Markup Language
UN	United Nations
UNESCO	United Nations Education, Science and Culture Organization
Unicode	Universal coding
UNIMARC	UNIversal MAchine-Readable Cataloging
UNISIST	Bibliographic standard developed by UN (France)
URL	Universal Resource Locator
UTF	Unicode Transformation Format
VAX	Virtual Address eXtension
VLIR/UOS	Vlaamse Interuniversitaire Raad/ Universitaire Ontwikkelings samenwerking
VMS	Virtual Memory System
Web-JISIS	Web-Java Integrated Set for Information Services
WHO	World Health Organisation
WINISIS	Windows Integrated Set for Information Services
WIS	Web Information Systems
WWW	World Wide Web
WWWISIS	World Wide Web Integrated Set for Information Services
Wxis	Web and XML enabled ISIS Server
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XPath	XML Path
XQuery	XML Query
XRF	Cross-Reference File

## List of Figures

Figure 1: ISIS Software Family .....	- 24 -
Figure 2: ABCD Suite .....	- 27 -
Figure 3: CGI Protocol Architecture .....	- 28 -
Figure 4: ISIS Master File Record Structure.....	- 30 -
Figure 5: Web-JISIS Three Tier Architecture.....	- 38 -
Figure 6: How use of NoSQL Flattens the Total System Cost and the Asymptotic Degredation of the Performance .....	- 41 -
Figure 7: Summary of Indexing Process in Lucene .....	- 50 -
Figure 8: Web-JISIS Login Form.....	- 55 -
Figure 9: List of Database Displayed from Web-JISIS .....	- 55 -
Figure 10: Database Record Display from Web-JISIS .....	- 56 -
Figure 11: Ethiopian PDF Loaded in J-ISIS, seen from the Data Viewer of Web-JISIS .....	- 72 -
Figure 12: Chinese PDF Loaded in J-ISIS, Seen from the Data Viewer of Web-JISIS .....	- 72 -
Figure 13: Chinese Document in Web-JISIS after Having Done a Search with Highlighted Search-keys	- 73 -
Figure 14: Illustration of J-ISIS Capable of Handling Mixing Scripts: Case of Amharic and English.....	- 73 -
Figure 15: Illustration of the New Added Field Types .....	- 78 -
Figure 16: J-ISIS Field Definition Table Contains Both Normal and Digital Library Field Types .....	- 79 -
Figure 17: J-ISIS Data Entry Worksheet Definition .....	- 79 -
Figure 18: Selection of Fields in J-ISIS Field Selection Table .....	- 80 -
Figure 19: Integration of the New Digital Library Feature and J-ISIS.....	- 81 -
Figure 20: Illustration of the Resulting Interface for Traditional and Digital Library Creation .....	- 81 -
Figure 21: File Browser for Uploading a Document into J-ISIS Worksheet .....	- 82 -
Figure 22: Document Content Loaded into J-ISIS Worksheet.....	- 83 -
Figure 23: Entering a Search Key into J-ISIS Search Interface .....	- 84 -
Figure 24: Illustration of the Search Result from J-ISIS Interface.....	- 84 -
Figure 25: J-ISIS with Menu Building Problem	- 86 -
Figure 26: J-ISIS with Correct Menu .....	- 86 -
Figure 27: J-UNIT Related Error Reported in Netbeans .....	- 86 -

## List of Tables

Table 1: ISIS Family Comparison.....	- 53 -
Table 2: Original vs Parsed Document Size Analysis .....	- 93 -
Table 3: Comparison of Greenstone Digital Library System with J-ISIS .....	- 97 -

## Abstract

*J-ISIS is UNESCO's new information storage and retrieval database software. It uses Berkeley DB for providing unlimited storage capability; Lucene for searching and retrieval of documentary resources; and Web-JISIS to provide web-based access to resources of the library. It is also Unicode compatible. Though it could be a good candidate for digital library applications, there was no any assessment regarding its capability and power for the use of digital library applications. In addition, there were no any digital library concepts and functionalities integrated with J-ISIS. In this study all technologies involved, including the limitations of the classic ISIS-technology for digital libraries, are discussed. An effort is made to test the already available features such as Web-based accessing of resources, full-text indexing, unlimited storage capability of the software and its Unicode compatibility. Moreover, a new interface that integrates Tika into J-ISIS is implemented, which allows to easily build digital library collections with the text content extracted and full-text indexed in the database and a hyperlink to the original document automatically provided. A comparison between J-ISIS and a well established digital library software, i.e. Greenstone, has been made. J-ISIS is found to be more efficient in terms of storage space and indexing speed. Finally, as a result of the tests conducted and the feature implemented, our conclusion is that J-ISIS can deal very well with digital library applications and UNESCO is strongly recommended to elaborate and promote this new approach.*

## **Chapter One: Introduction and Problem Statement**

### ***1.1. Introduction***

In this first chapter, we will present the background of the study focusing on the concepts and application of digital libraries. Moreover, we will define the problem statement; describe the general and specific objectives of this paper, its significance and the methodology that will be used to address the problem.

### ***1.2. The Digital Library Concept***

One of the crucial advances in the field of information technology and library systems is the digital library (DL) technology. It is a relatively new field in computer science, but there are many dynamics and continuous innovations in this area to improve the way information is accessed, collected and accumulated.

A digital library is a software system, mostly based on web-technology, which is used for storing and retrieving electronic documents giving access to the original documents, which in turn could be digital representations of artefacts of any nature or format, such as manuscripts, object images or native electronic documents e.g. website. These electronic documents can be stored in a wide variety of formats such as PDF, word-processor documents, XML, HTML, and multimedia materials (de Smet, 2011).

Digital library stores information as digital objects. These objects represent the content of a digital library and associated data which are called metadata. Due to this fact, digital objects are the building blocks of a digital library. In order to represent useful information, contents must have associated types such as text, image, audio, etc. In addition, each object in a digital library must have a unique identifier and it may also contain some additional information such as rights associated with the digital object, access methods and properties which indicates whether the digital object is mutable or not. For instance, if the object is mutable then it means the object can be changed after it is stored in a repository; also digital objects may have a digital signature which evaluates whether the object has been changed or not (Arms, 1995).

The digital library concept follows the principle of “what you store is not what you get”. This means, the digital contents available in the world are organized in many different ways and have to be accessed through a variety of mechanisms such as executing simple and complex programs on these objects. As a result, the underlying architecture must distinguish carefully digital objects created by the original creator, stored in a repository and digital objects as disseminated to the end user (Arms, 1995; Vijayakumar & Jeevan, 2001).

The digital library development process is quite an expensive task and it requires a lot of resources. Due to this fact, it will be important to consider some principles which should be followed in the design, implementation and maintenance stage of any digital library. The digital library which is designed with those principles in mind will benefit as it will be more usable and easily adaptable in a long term changes. The principles are mentioned as follows:

- knowing the content;
- involving the right people;
- designing usable systems;
- ensuring open access;
- be aware of data rights;
- automating whenever possible;
- adopting and adhering to standards;
- ensuring the quality;
- concerning about persistence

(McCray and Gallagher, 2001).

According to Candela et al. (2007), digital libraries represent a multidisciplinary field, including data management, information retrieval, library science, document management, information systems, the web, image processing, artificial intelligence, human-computer interaction, and digital curation. The evolution of digital library as a system provides an opportunity to digitized books and other text documents.

Lee et al. (2005) summarized the following major advantages of digital libraries over traditional libraries:

- digital libraries bring the libraries closer to the users;

- computer technology is used for searching and browsing;
- information can be shared;
- information is always available;
- new forms of information become possible.

The scope of digital libraries has expanded from basic storage and retrieval of information functions into much wider applications such as facilitation of communication, collaboration, and other forms of dynamic interaction among scientists, researchers, or the general public on themes that are related to the information stored in the digital library. Moreover, the capabilities of digital libraries are getting being advanced, serving from handling mostly centrally located text to synthesizing distributed multimedia document collections, sensor data, mobile information, and pervasive computing services (Candela et al, 2007).

We consider digital libraries as a complex concept, reaching from digitally accessible classic document collections to interactive collaborative information resources, bringing libraries fully into the modern computer age.

The main reason why libraries prefer digital collections are the following:

- digital journals can be linked from and to indexing and abstracting databases;
- access can be from the user's home, office, or dormitory whether or not the physical library is open;
- the library can get usage statistics that are not available for print collections; and
- digital collections save space and are relatively easy to maintain;
- when total processing and space costs are taken into account, electronic collections may also result in some overall reductions in library costs

(Tenopir, 2003).

Whereas the best known digital libraries, such as Europeana (<http://www.europeana.eu/portal/>) and Google Books (<http://books.google.com/>), are huge projects, in this thesis we focus on the technology for local smaller digital library applications.

### **1.3. Web Information Systems**

Besides searching and accessing resources, digital libraries have much wider functions than the web search engines. These functions include establishing context around those resources, enriching them with new information, and relationships that express the usage patterns and knowledge of the library community. That is, they will become a context for information collaboration and accumulation (Lagoze et al., 2005).

Digital libraries are becoming more web-oriented and sophisticated based on Web information Systems (WIS) technology. WIS are information systems which use Internet web technologies to deliver information and services to users or other information system applications. It is a software system of which the main purpose is to publish and maintain data by using hypertext-based principles.

Barna et al. (2003) identified the most important difference between WIS and traditional (non-web) information systems as the large amount of information organized in a web structure realized via (hyper) links available to a large number of potentially heterogeneous end-users. Thus, WIS needs solid approaches for conceptual structuring the information space and its access (often referred to as authoring) and for engineering and implementing the required access services. The need for fast and effective authoring and the increasing complexity of the systems ask for a rigorous and systematic design process.

According to Dempsey (2006), there are three important issues as far as the new web-oriented environment is concerned. These are: 1) Flat Applications and Liquid Content, 2) New Social and Service Affordances, and 3) New Business and Organizational Patterns. He also explained these three characteristics as follows:

- The world is moving to a flatter network world, where the gap between the Web and business applications is narrowing. Applications are working over the Web. Data are flowing more readily into user environments. Web services and RSS are important parts of a spreading connective tissue which allows users to compose services in different environments. In this context, workflows and business processes are being further automated, data are more accessible and malleable, and applications may be more flexibly built and reconfigured from components.

- Flat applications and liquid content create new opportunities:
  - 1) *Workflow and process standardization* allows organizations to think about how best to source activities, perhaps outsourcing some activities so as to concentrate on where they most add value;
  - 2) flatter applications and liquid data allow greater collaborative working, maybe through sharing of components, collaboratively sourcing shared activities, or working on shared problems; and
  - 3) great upsurge in *social networking services*, where a flat connective tissue is based on blogs, wikis, IM (Instant Messaging) and other tools to create social and communication spaces in which new services are being built.
- Finally, Business models and organizational structures are co-evolving with the technical developments and new services that they allow. This includes the way in which *information resources* are flowing onto the network, free at the point of use or available for a small fee; *On-demand and platform services* supported by automated workflows and process standardization are becoming more viable options; and this continued interdependence in a flatter world means that services are increasingly *co-created*, and this co-creation is extending to the relationship between a service provider and its users. Moreover, the web environment has characteristics of comprehensive, integrated D2D, making data work hard, horizontal platforms and interfaces, and a co-created experience.

The benefits of Web Information Systems are immense and multi-dimensional. It is used in almost every sector, provided that there is the necessary technology infrastructure.

According to Meyer et al (2007), WIS are extensively used in managing and disseminating cultural heritage data. Safeguarding and exploiting cultural heritage induces the production of numerous and heterogeneous data. The management of these data is an essential task for the use and the diffusion of the information gathered on the field. Previously, the data handling was a handmade task done thanks to efficient and experienced methods. Until the growth of computer science, other methods have been carried out for the digital preservation and

treatment of cultural heritage information. The development of computerized data management systems to store and make use of archaeological datasets is then a significant task nowadays. Especially for sites that have been excavated and worked without computerized means, it is now necessary to put all the data produced on computer. It allows preserving the information digitally (in addition with the paper documents) and offers new exploitation possibilities, like the immediate connection of different kinds of data for analyses, or the digital documentation of the site for its improvement.

#### ***1.4. Some Technological Requirements for Digital Libraries***

In order to accomplish the function that we have seen above, digital libraries need some typical technical requirements such as a multitude of meta-data sets, full-text retrieval, and compatibility with the Open Archives Initiative standards.

- Digital libraries are special information systems, due to their efforts to create added-value (like traditional libraries do) by describing the documents in the collections with more or less standardized sets of attributes ('fields'), called 'meta-data', which are in fact indeed very similar to bibliographic structures of the classic library approach.
- Another typical requirement for digital libraries is the capability to allow retrieval of documents based on full-text, meaning not only the added meta-data act as entry-points for searching and retrieval, but all words within text-documents also are indexed and act as search keys.
- Technical compatibility with OAI-PMH is an important characteristic of digital libraries. The Open Archives Initiative develops and promotes interoperability standards that aim to facilitate the efficient dissemination of content. The work of OAI-PMH has expanded to promote broad access to digital resources for eScholarship, eLearning, and eScience (de Smet, 2010).

#### ***1.5. ISIS and ABCD for Digital Library Applications***

The ABCD integrated library system is one example of a software which is fully based on ISIS-database technology. This ISIS technology has suitable technical characteristics required for the application of digital library functions, although with real limits. We will investigate in detail to this ISIS technology as part of this paper.

According to de Smet (2010), these technical characteristics, relevant for digital library applications include among others 1) freedom of database structure; 2) full-text indexing; Since ABCD is based on ISIS for the data-storage and -retrieval on the one hand, and PHP for the web-interface creation on the other hand, it benefits from both technologies. E.g some nice PHP-tools are embedded, allowing editing of full-text content within the records with the FCKEditor-tool; However, the classic ISIS-technology has some limitations such as Unicode compatibility, and record- and field-size limitations.

To solve these limitations, an initiative is envisaged to update the ABCD integrated library system based on J-ISIS. J-ISIS is developed by UNESCO in 2008 as a Java-based graphical interface and it uses Java technology and the embedded Berkeley DB for the storage layer. As with all ISIS-related software, this project is a fully Free and Open Source Software (FOSS)-oriented project. It is the result of continuous effort to improve the data storage capacities and consistency with the modern technology of the ISIS family (de Smet, 2009).

According to Dauphin (2008), J-ISIS is a successor of WinISIS. He stressed that it is being developed thanks to collaboration with the University of Antwerp, and that UNESCO maintains its position with the worldwide user community and the continuity of support to libraries in developing countries. J-ISIS was developed from the beginning with Unicode in mind and under the Open Source philosophy. J-ISIS has adopted technologies based on Berkeley DB for the management of records of variable length, as well as Lucene for indexing and data retrieval, while maintaining the typical ISIS-family formatting language (a parser to select data from the databases and present them in different formats). He also commented that UNESCO is working on an agreement with SUN Micro Systems for further development and financing for the project.

In general, de Smet (2009) has summarized the advantages of this new ISIS-technology as follows:

- It will have a flexible architecture in which 'ISIS-cells' will communicate through known protocols with several platforms and interfaces; ISIS-cells will also allow to use different storage models as these will be contained within the cells but they behave in the same standardized way towards the external technology used;

- The new ISIS generation will avoid the limitations related to database-, record- and field-sizes;
- ISIS databases will be Unicode compatible; and
- It will use Lucene, a FOSS software from Apache Software Foundation for full-text indexing.

The new products of the ISIS family, and UNESCO's commitment to apply the same license to the new generation of ISIS, will affect the evolution of habits and roles of actors in the ISIS environment (UNESCO, BIREME, National Distributors, application developers and final users).

The ISIS Software family has a unique technological concept and developmental mission to cope with Information Storage and Retrieval Systems (ISRS), particularly for developing countries where the technology is widely known and used. The new developments led by UNESCO (especially the J-ISIS project showing renewed commitment) and BIREME (especially the Network Based Platform and the integrated library management software ABCD) represent important advancements toward the updating of the ISIS platform according to the state of the art in software development as applied to information storage and retrieval systems (BIREME, 2008).

Though J-ISIS was introduced in 2008, there have not been practical tests made to ensure the reconciliation with new technology and modernization, its compatibility with previous programs, and its storage capacity and user friendly advantages. Therefore, the aim of this study is to test the new ISIS environment for digital library capabilities with attention to both User Interface Design and system performance. An experimental implementation as a Web Information System using JSP and Struts, i.e. 'Web-JISIS', will also need to be tested and described.

### **1.6. Objective of the Study**

The general objective of this study is to test the hypothesis that the new ISIS technology indeed allows full implementation of digital library technology by analyzing the feasibility of a prototype implementation of such digital library functions into the J-ISIS-based information system.

The specific objectives include:

- Describe the classic ISIS database technology from a technical point of view;
- Describe the NoSQL database philosophy as implemented by Berkeley DB;
- Assess Lucene full-text indexing;
- Assess the current J-ISIS (Java based ISIS) and Web-JISIS implementations;
- Implement some features, such as easy use of multiple document formats and their conversion to TXT/HTML for inclusion into the databases;
- Digital library technology will be tested in J-ISIS by loading large documents into J-ISIS records, and some performance measurements on indexing and retrieval will be done, also in comparison with a well-known dedicated digital library software (Greenstone).

### **1.7. Significance of the Study**

Since there are no practical tests made yet in this area, the study will be a first attempt to cover this problem. Moreover, the result of the study could be used as a report to UNESCO about the possibilities and requirements for further development of J-ISIS by adding digital library functions.

Such report could give input to the discussion on the interest in and importance of digital libraries, especially as seen by UNESCO in view of its mandate to defend local information production/management and cultural heritages, and some technological requirements for software supporting these goals.

### **1.8. Methodology**

The methodology used in this thesis to address the problem is very similar to the 'Design Science Research' approach which is well accepted in Information Systems research (Koechler, 2011). We have analyzed the problem and design a prototype solution for it, which we have tested and evaluated. This is based on the constructivist principle of 'learning through building' with an emphasis on relevance to practice. Therefore, all elements of the problem (the technological requirements for digital libraries as we see them) are discussed and the solutions are tested accordingly; and build a new one where the solution was missing (an interface for building a digital library collection) with practical application.

## **Chapter Two: Discussion of Some Technical Requirements of Digital Libraries**

In this chapter, a brief discussion on some of the technical requirements of digital library technology will be presented. To be specific, the discussion will focus on web-based retrieval, the diversity of document formats, the diversity of metadata formats, full-text indexing and ranking, Unicode support, and as additional desired features: OAI-PMH and interactivity.

### ***2.1. WWW-based Retrieval***

One of the technical requirements of digital libraries is web-based retrieval of digital library resources. This makes digital libraries more interactive and accessible. As a result, users of digital libraries can interact with the system efficiently and access the system at anytime from anywhere.

“Information Retrieval (IR) is the area of study concerned with searching for documents, for information within documents, and for metadata about documents, as well as of searching databases in the WWW. There is overlap in the usage of the terms regarding data retrieval, document retrieval, information retrieval, and text retrieval, but each also has its own body of literature, theory, praxis, and technologies” (Wikipedia, 2011).

Internet analogy of public libraries i.e. reliable, high-quality community services have only recently begun to appear. This is because digital libraries are expensive to create and maintain. A serious obstacle to their creation is the provision of appropriate cataloguing information. Without a database of titles, authors and subjects, it is hard to offer the searching and browsing facilities normally available in physical libraries. Full-text retrieval provides a way of approximating these services without a concomitant investment of human resources (Witten et al., 1998).

According to de Smet (2011), the web-based environment has enabled full-text retrieval of documents as a standard feature. A digital library software, therefore, will not only offer the resulting descriptive data (the meta-data records) from a search, but also the full documents themselves from which search keys might also have been used for retrieval. This full-text retrieval feature of digital library software offers major advantages to physical libraries (and

their software) which are mostly only indexed by their bibliographic (or meta-data) descriptive data.

## **2.2. Diversity of Document Formats**

The term [electronic] document is used to denote any information bearing a message in electronically recorded form. In a digital library, a document is a particular electronic encoding of what in library science is called a “work” (Witten et al., 2002).

According to FAO and UNESCO (2005), an electronic document is a digital representation of ideas or creative or intellectual works which are logically complete and can exist on their own as an independent unit of work. Generally, electronic document formats can be grouped into three types: text based formats, image formats, and audio and video (multimedia) formats. The usability of electronic documents depends on several factors, e.g.: the document format and related mark-up codes (procedural mark-up, presentational mark-up or descriptive mark-up). Document formats have many characteristics that need to be considered when deciding on the suitability of any format. Electronic documents consist of a combination of text and images. Text is encoded using a variety of conventions, now converging towards the Unicode system.

Witten et al (2002) summarized the four principal styles of format in which electronic documents are expressed as follows:

- *The first comprises in web oriented formats such as HTML, XHTML and XML document manifestations such as the Text Encoding Initiative and Open eBook format. The principal difficulty here is that such documents are not always self-contained: they frequently include explicit links to other resources such as images or other documents.*
- *The second style of expression comprises word-processor formats such as Microsoft Word or RTF (“rich text format”). RTF is designed to allow word-processor documents to be transferred between applications, and uses ASCII text to describe page-based documents that contain a mixture of formatted text and graphics. In contrast, the native Word format is intended for use by a single word processor. Strictly speaking, it is inappropriate to use this format to convey documents to digital libraries; nevertheless, users often want to do that.*

- *The third style of expression for documents comprises page description languages like PostScript and PDF. These combine text and graphics by treating the glyphs that express text as little pictures in their own right, and allowing them to be described, denoted, and placed on an electronic “page” alongside conventional illustrations. They portray finished documents, ones that are not intended to be edited, and are therefore more akin to traditional library documents than word-processor formats. Most of the time digital libraries can treat documents in these languages by processing them using standard “black boxes”: generate this report in a particular page description language, display it here, transfer it there, and print. However, to build coherent collections out of the documents, it is beneficial to be able to extract the text for indexing purposes and some elements of document structure for browsing purposes, and these are challenging problems.*
- *The fourth style of expression is media-rich documents such as sound, pictures and video. When accompanied by textual descriptions (the view taken here), their treatment becomes one of associating metadata with documents. This provides a baseline approach that unifies the different media types and permits all the metadata methods discussed below for the general case to be applied.*

### **2.3. Diversity of Metadata Formats**

The term ‘metadata’ is used to represent structured data that describes a resource in the collection of a library, manages rights about the resources, identify and discover resource in a collection and also it provides preservation of a digital resource. In addition, the use of metadata standards in a digital library enhances interoperability (Singh, 2003).

Metadata is very important to ensure that resources will survive and continue to be accessible into the future. It also facilitates an easy way of searching and retrieving of resources. For instance: it provide data about data, time and date of creation, creator or author of data, placement on a computer network where the data was created, and standards used (NISO, 2001).

Digital libraries describe all the documents in a collection using a set of standardized attributes i.e. metadata. One can say that digital libraries are more than just information systems, due to their efforts to create added-value (like normal libraries do) by describing the

documents in the collections with more or less standardized sets of attributes ('fields'), called 'meta-data', which are in fact indeed very similar to bibliographic structures of the classic library approach (de Smet, 2010).

The best-known standard for metadata is the Dublin Core (see <http://dublincore.org>), suggesting only a core set of crucial fields to describe a very wide variety of information objects. Many other standards are based on this or derived from it, but all follow the same idea, i.e.: characterizing the collection objects by content, creation, responsibility but e.g. also, and interestingly, by management aspects like technical requirements for access and preservation.

The capability to deal with different metadata formats, or 'metadata openness', is a very important technical requirement for digital libraries. Openness in metadata definition implies that there can be different data structure specifications to be used based on the need of the application going to use the metadata, without imposing a fixed structure on the real-world objects.

## **2.4. Full-text Search and Ranking**

Another crucial technical requirement for digital libraries is the capability to allow resources retrieval based on full-text of the document, meaning not only the added meta-data act as entry-points for searching and retrieval, but all words within text-documents also are indexed and act as search keys. The internet-world dominated by Google has made such things obvious and enabled such functions easily while such functions were uncommon until recently (de Smet, 2010).

### **2.4.1. Full-text Search**

Full-text Search (FTS) is a search for the documents, which satisfy a query statement specifying a word to be present in the document and, optionally, return them in some order. It finds the documents containing all query terms and returns them in order of their similarity to the query. This is the most usual case of FTS functions. These notions of query and similarity are very flexible and depend on specific applications. The simplest search machine consider query as a set of words and similarity - e.g. how frequent are query words in the document (Bartunov and Sigaev, 2007).

According to de Smet (2010), the full-text retrieval feature of digital library software offers a major advantage to physical libraries (and their software) which mostly is only indexed by their bibliographic (or meta-data) descriptive data. Thus, a digital library software offers not only the resulting descriptive data (the meta-data records) from a search, but also the full documents themselves from which search keys might also have been used for retrieval. Moreover, Bartunov and Sigaev (2007) stressed the fact that modern information systems are all database-driven and there is a need in IR-style full-text search inside databases with *full conformance* to the database principles ACID (Atomicity, Consistency, Isolation, Durability). That's why many databases have built-in full-text search engines, which allow combining text searching and additional metadata, stored in various tables and available through powerful and the classic SQL language.

#### **2.4.2. Ranking**

Ranking is about presenting search results with most relevant documents first, which can only be offered in digital libraries, not classical libraries, because there is control of the full-text. E.g. page ranking of Google adds 'social usage' parameters (the number of times users have followed a link to the document) to the statistical parameters simply counting the frequency of words occurrence. Especially in large collections where typically many results are given, ranking is a useful feature.

#### **2.5. Unicode**

Unicode was developed to provide a single encoding for all of the world's major language scripts or alphabets. It provides a unique number for every character (Oracle, 2009). It is an encoding scheme which is developed by the Unicode Consortium (incorporated under the name Unicode, inc. in 1991) and the ISO (International Organisation for Standardization). The Unicode Consortium is backed by most of the major players in the IS game, including Adobe Systems, Apple Computer, Compaq Computer, etc. In 1992, the Unicode Consortium and ISO agreed to merge their character encoding standard, so the character sets map exactly. Thus, in addition to assigning names and bit-pattern mapping to characters (in conjunction with the ISO) the Unicode standard also provides implementation algorithms, properties and semantic information. The basic original premise was to use 16-bits for every character which allows for 64K unique patterns (65,536) and maintains the compatibility with as many already existing standards as possible (Comstok, 2006).

The old standard code pages have the following disadvantages:

- each covers only a subset of all characters used;
- incompatibility between different code pages; and
- only restricted data exchange possible.

However, Unicode solves all these problems. Thus, Unicode is an important technical requirement of digital libraries since it provides one code page for all scripts. Using Unicode, more language scripts can be supported easily without the need for new code pages or other new methods (Hansen, 2005).

According to Hansen (2005), the Unicode Standard was adopted by IBM and several other companies including Apple, HP, JustSystem, Microsoft®, Oracle, Sun™, Sybase, and Unisys. Moreover, Unicode is required by modern standards such as XML, Java™, ECMAScript (JavaScript™), LDAP, CORBA 3.0, and WML. Unicode is also the official way to implement ISO/IEC 10646 and is supported in many operating systems and all modern browsers.

Comstock (2006) has summarized the three alternative ways of representing Unicode characters as follows:

1. *UTF-16-the basic 16-bit encoding scheme: two bytes used for every Unicode character.*

*But version 3.0 of the Unicode standard introduced a concept called surrogate pairs that allows some Unicode characters to be represented by a pair of two-byte values.*

2. *UTF-8- an algorithm for converting Unicode characters to a string of characters that are one, two, three, or four bytes in length, and back.*
3. *UTF-32- a 32-bit encoding, the basis for the ultimate character encoding, allowing for 1,114,112 character assignments (note: the leftmost 11 of the 32 bits must be all binary zeros)*

✓ *This encoding was made an official part of the Unicode standard in version 3.1 in May, 2001.*

In general, due to the fact that digital library needs to handle different types of resources such as documents, cultural heritages, artefacts written in different local languages, making the digital library to support Unicode plays an important role.

## **2.6. Open Archives Initiative-Protocol for Metadata Harvesting (OAI-PMH)**

There are a lot of efforts made by many educational institutions and international organizations such as UNESCO to avail their documents, theses, dissertations, articles, artefacts and other cultural heritages through digital library technology. However, there is a lack of interoperability between these libraries.

The critical importance of OAI-PMH here is to develop and promote interoperability standards that aim to facilitate the dissemination of content. One key project is the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH, <http://www.openarchives.org/pmh/>) which provides a “low-barrier mechanism for repository interoperability” for archives (institutional repositories) containing digital content (digital libraries). OAI-PMH allows people (service providers, such as the ones registered with the OAI, listed on <http://www.openarchives.org/service/listproviders.html>) to harvest metadata (from data providers, such as the ones registered with and validated by the OAI listed on <http://www.openarchives.org/Register/BrowseSites/>). While Data Providers administer systems that support the OAI-PMH as a means of exposing metadata, Service Providers use metadata harvested via the OAI-PMH as a basis for building value-added services (Hornik, 2009).

The OAI-PMH protocol enables digital libraries to provide access to the documents stored to external users, which allows accessing of digital library resources without entering into the proper system and search software. Thus, the technical compatibility with OAI is an important characteristic for digital libraries in order to provide a server which can receive and understand the limited set of agreed requests (GetRecord, Identify, ListIdentifiers, ListMetadataFormats, ListRecords and ListSets) and to be able to return XML-formatted responses based on the OAI-PMH Scheme (de Smet, 2010).

The main goal of OAI-PMH is to develop a low-barrier, lightweight framework to facilitate the discovery of content in distributed archives which supports data providers (repositories, archives) and service providers. The service providers develop value-added services based on the information collected from data providers whereas data providers are simply collections of harvestable metadata that may or may not contain additional services and content (Sathish et al., 2009).

There are a lot of characteristics which makes OAI-PMH to be used widely. These characteristics include the fact that OAI-PMH is harvest-oriented, hence its focus is on the transfer of metadata between participants, and common services like searching are outside its scope. Services that add value to the metadata may be implemented by Service Providers. There are two harvesting criteria: by date stamps (for creation, deletion or modification) and sets, collections of records defined by the provider. Another characteristic is its simplicity, the requests are simple HTTP GETs or POSTs and the responses are given in an XML format defined by the OAI. It supports multiple formats, but Dublin Core is mandatory for interoperability. It is scalable because there is no processing of metadata and access to the data is asynchronous (Reis and Freire, 2008).

The development of OAI-PMH is driven by several factors. Some of these factors include the fast growth of the Internet, the slow traditional publishing model for the scholarly fields, the transfer of rights from author to publisher blocking the dissemination of results, the peer-review working as a barrier to new ideas as articles from prestigious institutions are favoured and subscription prices to publications become too high for libraries. The initial OAI target repositories were e-print archives, where author-submitted research papers are stored. The first prototype of OAI-PMH resulted from the Santa Fe Convention held to discuss and try to solve the problems described above (Reis and Freire, 2008).

There are some terminologies that need to be defined in relation to OAI-PMH. Reis and Freire (2008) have summarized these terminologies as follows:

*1. Participant Types:*

*As intended in the Santa Fe Convention, two participant types were defined: Data Providers, which are entities that support the OAI-PMH as a means of exposing*

*metadata; and Service Providers, which use the harvested metadata to build value-added services. Some systems may act as both.*

- ✓ *Data Providers: Data Providers refer to entities who possess metadata and are willing to share this with others. These entities provide free access to metadata, and may offer free access to full-texts or other resources (but that will be out of the scope of the protocol). One of the goals of OAI-PMH is to provide an easy way to implement a low barrier solution for Data Providers.*
- ✓ *Service Providers: Service Providers are entities who harvest data from Data Providers in order to provide higher-level services to users. Typical services are searching, browsing, etc. An interesting scenario is when Service Providers can process the data and use it to provide services as Data Providers (such as, for example, harvesting all the records from a group of libraries and provide them again regrouped in new sets mixing records from more than one source).*

## 2. The OAI-PMH Software

- ✓ *Harvester: A harvester is a client application that sends OAI-PMH requests. It is operated by a Service Provider as a means of collecting metadata from repositories. It is the mechanism Service Providers have to retrieve data from Data Providers.*
- ✓ *Repository: A repository is a server accessible in a network, able to process OAI-PMH requests. It is managed by a Data Provider to expose metadata with the purpose of being harvested. To explain what is the Repository there are three concepts that must be described as well:*
  - *Resource – what the metadata refers to, the nature of which is outside the scope of the OAI-PMH.*
  - *Item – a constituent of a repository from which metadata about a resource can be disseminated.*
  - *Record – metadata from an item, in a specific metadata format, encoded in XML.*

- ✓ *Metadata Formats: OAI-PMH makes use of the metadata of resources (ex: a PDF file), and not resources themselves. Below the resources are items, the most abstract entity in OAI-PMH. An item has a unique identifier and is the entry point for all the resource metadata. Below the item are records. Records have unique identifiers and contain the metadata in a specific format in XML: MARC, Dublin Core, Qualified Dublin Core, MODS, METS, the European Library application profile for objects (TEL-AP), etc.*
  
- ✓ *Identifiers: An identifier in OAI-PMH must be unique so it can unambiguously identify an item within a repository. The unique identifier maps to an item, and all possible records available from a single item share the same unique identifier. The format of the unique identifier must correspond to that of the URI (Uniform Resource Identifier) syntax. The scheme component of the unique identifiers must not correspond to that of a recognized URI scheme unless the identifiers conform to that scheme. Unique identifiers play two roles in the protocol: in the response for the verbs ListIdentifiers and ListRecords to identify each record; and to request a specific record in the request GetRecord.*
  
- ✓ *Repository Sets: Sets are an OAI-PMH mechanism to allow for harvesting of sub-collections, whose semantics are defined outside of the protocol. Sets are defined by conventions established between Data and Service Providers, or just by the Data Provider. They may be several logic aggregations, like: Journal issues, Institutional Repositories, EPrint Archives and Collections with some kind of associated semantics. In The European Library an OAI Set will be available in the portal as a collection. As such, it will have a collection description in the portal.*

Finally, it is important to say something about the OAI-PMH related tools that can be used either to build a digital library from scratch or customize an existing digital library. While on the data provider side, application packages (e.g. Dspace) provide a complete software for building a data provider with an integrated OAI-PMH module, on the service provider side, the off-the-shelf application packages (e.g. CDSware) and development frameworks can be used. An existing digital library can be made OAI-PMH compliant by adding a software layer, which receives OAI-PMH requests over HTTP, interacts with the digital library to

fetch the requested information and sends out the results. Typically, this interaction involves fetching the required metadata from the underlying storage scheme followed by mapping of the metadata to mandatory Dublin Core (DC) format required by OAI-PMH before sending it out. As this software layer needs to understand the underlying DL implementation, it tends to become specific for a DL. Large organizations can possibly use development frameworks such as NCSA Cocoa since the development of such software is not a large investment. However, small organizations would prefer a common tool that can be configured to make their small collection OAI-PMH compliant. For creating a new OAI-PMH library from scratch, one can use application packages like EPrint from Southampton. However the EPrint software, because of its installation and maintenance complexity, is suitable mostly for large organizations (Sathish et al., 2009).

## **2.7. Interactivity**

Another technical requirement that has been identified as relevant for digital libraries is interactivity. This technical requirement has been ignored in the previous works whereas it is a quite interesting phenomena and benefit as it creates an open space for discussion, giving feedback on the resources and improving the content quality of resources of a digital library. Implementing interactivity operation on digital libraries enables users to build knowledge on top of others' knowledge by sharing their knowledge. In this case, users will not be only readers of the resources published by someone but also can be publisher (e.g. giving a comment on the available resource). As a result, users will have a power of participation on the contribution of their knowledge to the content of existing resource. In traditional libraries 'reader evenings' are organized to discuss the readers opinions about a specific book or topic and share views: the library becomes also a social agent in knowledge sharing and creation. In the digital library this role could be performed by the interactivity feature of the system.

## **Chapter Three: History and Technical Concepts of the Classical ISIS Technology and ABCD Software**

### **3.1. Introduction on ISIS**

In this chapter, the history and technical concepts of the classical ISIS technology and the overview of ABCD (as the most advanced representative of classical ISIS-technology) will be discussed. In the first part a brief discussion of the historical background of the ISIS-generations will be presented. In the second section an overview of the ABCD software will be discussed. Finally, four technical concepts related to ISIS technology and ABCD software will be presented: the database storage engine and ISO 2709, the ISIS B-Tree Inverted File; the ISIS-Query language; and the ISIS-Formatting Language.

### **3.2. Brief History of the ISIS Technology**

CDS/ISIS is a “generalized information storage and retrieval system” software package which is used for creating and manipulating textual databases (Jayakanth, 2001), which are ideal to use for the catalogues in small and medium sized libraries as textual databases are well suited for bibliographic applications (Buxton, 2002). CDS/ISIS is the name of the ‘text-retrieval database’ software which was originally developed at ILO (Geneva) in the seventies and taken over for wider support and free distribution by UNESCO since 1985 (de Smet, 2008b).

According to de Smet (2010), the ISIS software family is a set of textual database softwares, based on the same data formats, to which a very strong retrieval engine (for searching information) and an even more powerful formatting language (for displaying/output of information) is applied. These data formats allow textual information of the ‘semi-structured’ type to be stored in a highly economical and organized way. All texts are stored in ‘fields’, and fields can have an undefined number of occurrences (zero, one or more), an undefined length (between the limits of - depending on the ISIS-member - about 8 to 64 KB, with one special version allowing up to 1MB), and can be further subdivided into subfields. Moreover, the ISIS software can accommodate for the international information formats MARC, CCF and other ISO 2709 based formats, characterized by large numbers of fields and special ideas

like subfields, which are most widely used in libraries, documentation and information centres.

When we trace back the history of CDS/ISIS software, it was created for the Central Library of the ILO in Geneva to process the abstracts of documents and was adapted further for library services within the ILO (Hopkinson, 1996). ILO decided to make the software and the source code available to other organizations in 1975. Accordingly, UNESCO and the International Development Research Centre (IDRC) in Canada adopted the software. Moreover, UNESCO developed the ISIS software further for its Computerized Documentation Service (CDS) to meet its own needs (Rodriguez, 1995). The product of this development was the first version of CDS/ISIS designed for an IBM mainframe computer which was released in 1975. Later on ISIS was developed into MINISIS by the IDRC (International Development Research Centre, see <http://www.idrc.ca>) which was designed for Hewlett-Packard minicomputers and distributed freely to developing countries (Mishra, 2006; Rodriguez, 1995).

In 1985 UNESCO released the Micro CDS/ISIS version which was designed to be used on the IBM Personal Computer. UNESCO has also been distributing the software freely to developing countries (Mahmood, 1998:23; Rodriguez, 1995:227). UNESCO's intention in developing CDS/ISIS was to benefit developing countries and hence promote the exchange of information between similar sectors or agencies (Hopkinson, 1996). With the launch of the Windows version, commonly called WINISIS, in 1998, the DOS version was suspended. The latest Windows version 1.5 appeared in 2003 (Abboy and Hoskins, 2008).

UNESCO not only provided the very successful 'Micro CDS/ISIS' version for DOS/PC but also the 'WinISIS' version for Windows (from 1997) with tens of thousands of (institutional and individual) users all over the world, mainly in developing countries. New versions and applications based on the ISIS-standard(s) such as a graphical development library ISIS\_DLL and web server software (WWWISIS and WXIS) have been developed in the environment of other UN-organizations such as FAO (e.g. with the WEBAGRIS and WEBLIS software) and WHO/BIREME (de Smet, 2008b).

As several institutions and individuals have programmed new variants of ISIS based on these same standards (with only minor deviations), de Smet (2008b) has summarized them as a ‘family’ of softwares: the ISIS-software family.

Thus, many applications have been developed based on these variants of ISIS which range from the ‘old-fashioned’ but very robust DOS- and UNIX-applications to rich graphical (Windows-based) end-user applications and powerful web-based applications (e.g. the earlier mentioned WEBLIS system as an integrated web-based library management software). These applications, along with the basic software, are widely used, especially in Latin America, but also in non-profit environments in Europe and the South (Asia, Africa) in general (de Smet, 2008b).

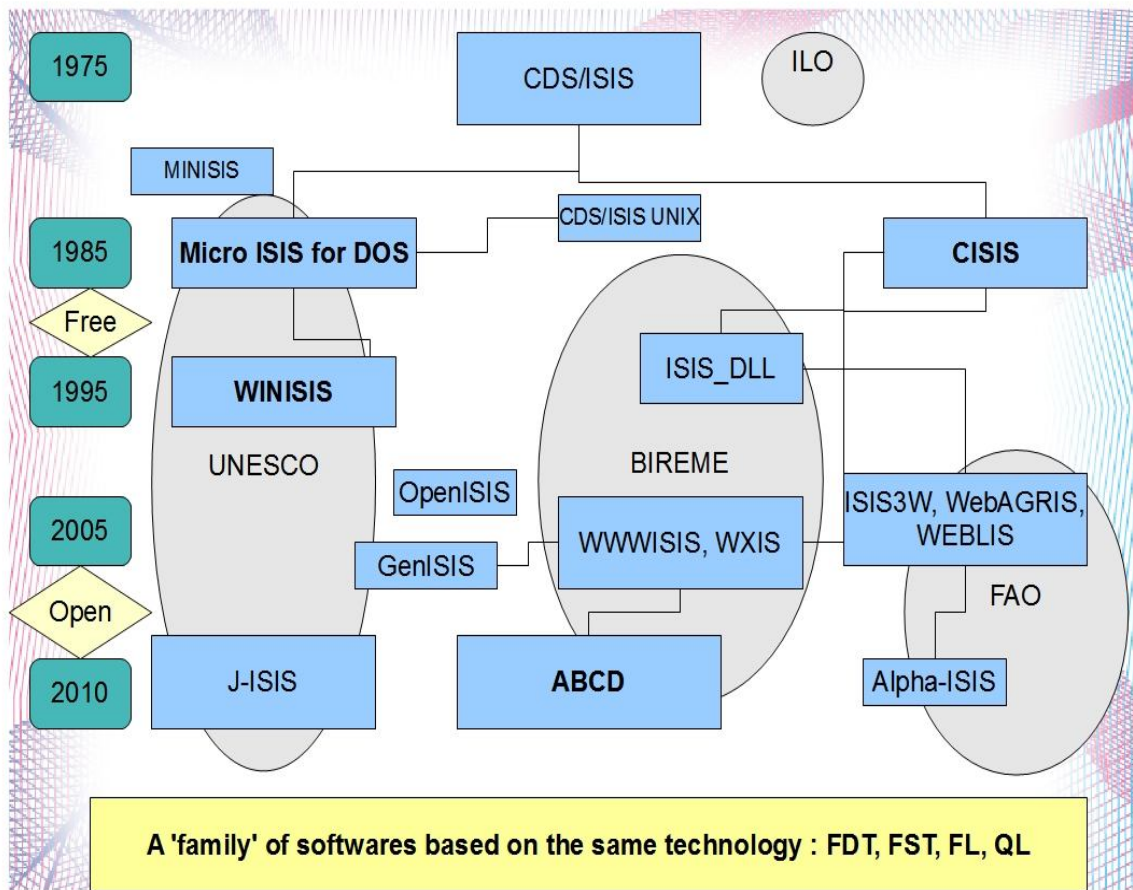
### **3.2.1. The ‘ISIS-software family’ in Four Generations**

Software applications associated with CDS/ISIS are so varied and numerous that it is sometimes referred to as a “family of software” (Matovelo and de Smet, 2005).

According to de Smet (1999), ISIS software comes with different versions:

- CDS/ISIS, a menu-based character-mode system on different computer platforms :
  - Micro CDS/ISIS for PC (MSDOS)
  - CDS/ISIS for UNIX (Intel based UNIX: SCO, Linux, Free BSD...)
  - CDS/ISIS for VAX (not maintained anymore)
  - CDS/ISIS comes with a built-in Pascal programming language.
- WINISIS, the Windows-based release of CDS/ISIS with multimedia extensions
- CISIS, a set of command-line tools for database maintenance, available for most UNIX-environments and DOS
- WWWISIS, a server software allowing for WWW based access to ISIS-databases on UNIX, NT (as a DLL) and DOS/Win95
- ISISDLL, a programming library to develop graphical interfaces and applications based on the ISIS-formats for data storage, retrieval and formatting
- JAVAISIS, a Client-Server software which allows you to browse any CDS/ISIS databases through a JAVA interface.

The first two family members are developed and maintained by the General Information Programme of UNESCO (Paris) for which the distribution is for free through accredited national distributors. The subsequent three software categories are developed and maintained by BIREME, Sao Paulo (Brazil), of which the distribution is either for free or for a moderate fee (which entitles for support) through the Internet. The last member, JAVASIS, was developed by the Documentazione le Biblioteche e gli Archivi (DBA), Italy; this software allowed client-server database access using Java but is now out of use and has nothing to do with the new J-ISIS except of using ISIS in a Java-context. All ISIS-software are deliberately multi-lingual (or rather: language-independent) and adaptable in nature, allowing several 'profiles' and interface-types (de Smet, 1999).



Source: de Smet (2009)

**Figure 1: ISIS Software Family**

As can be seen from the above figure, for simplicity, the family is organized into five generations the involvement of some UN organizations is indicated with two main streams: the UNESCO stream leading to the new J-ISIS and the BIREME stream leading to ABCD and the new 'Network Based Platform'.

According to de Smet (2008a), the new generation which can be termed as the fifth generation is characterized by the use of new technologies in data storage (Berkley DB in J-ISIS, Couch DB in ISIS-NBP, MySQL in Alpha\_ISIS), as well as by the full incorporation of Web technologies (such as PHP and JavaScript in ABCD, Plone CMS in ISIS-NBP), while everything is conceived in the spirit of FOSS development. As a consequence of all this, the ISIS master file (MST) and Cross Reference File (XRF) file architecture of previous generations will tend to disappear, while the formatting language will still be available for data management.

De Smet (2008a) considered that future developments of ISIS will no longer be classified simply as Database Management Systems, but will become the de facto standards for the management of textual, semi-structured databases. This new generation will maintain ISO 2709 as a common data exchange format and will be enriched with XML as a storage and exchange format. They will all use Lucene indexing and will all support Unicode.

### **3.2.2 The ABCD System Based on Classic ISIS Technology**

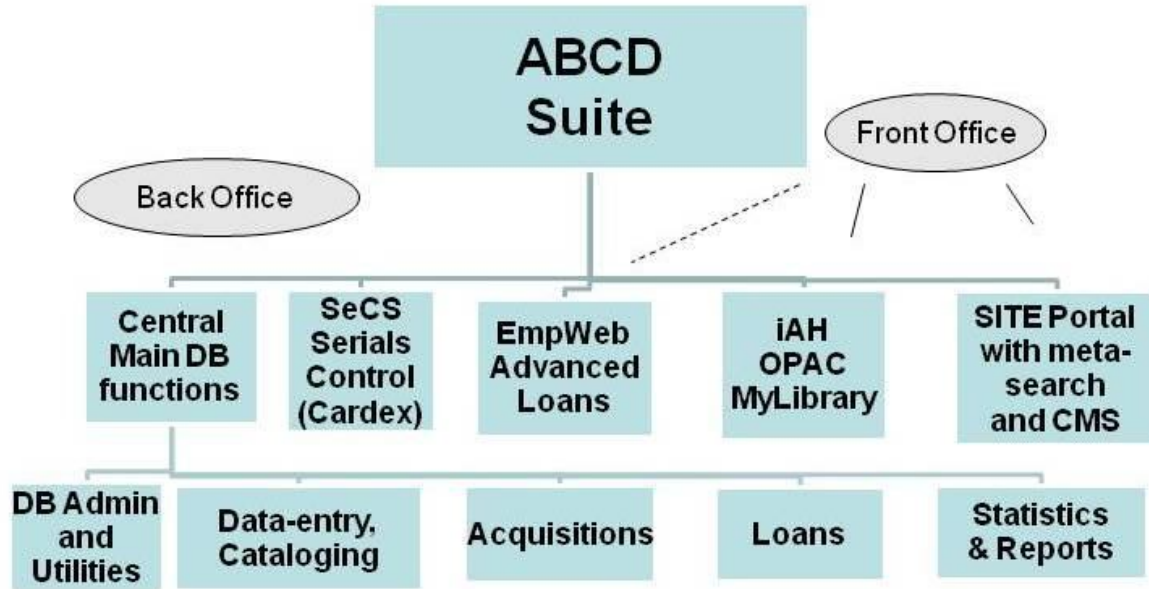
Following the substantial international development effort lead by BIREME (WHO, Brazil) and the Flemish-Belgian VLIR/UOS (inter-university development co-operation body), the first release of ABCD as an integrated library automation software based on ISIS-technology was published in December 2009. The software is provided as a 'FOSS', i.e. 'free and open source software', keeping the 'free' nature which has always been the case for ISIS which is now driven by officially adopted 'open source' philosophy in the United Nations, particularly UNESCO and BIREME (the two institutions where most of the ISIS-programming development is done) (de Smet, 2010).

The software entails, as a 'software suite' like e.g. Microsoft Office, typical library oriented modules for database administration (definitions of storage and indexing structures, data-

entry, searching, import-export etc.), loans administration, serials management, online end-user searching (OPAC) and portal with integrated meta-search and Content Management System (de Smet, 2009). Moreover, the software aims at facilitating automation of libraries, from documentation centres for specialized fields to big university libraries. The ABCD software has further-reaching ambitions: not only to automate libraries ('Automatisación de Bibliotecas' or AB in many languages) but also to provide a tool for automating Documentation Centres ('Centros de Documentación' or simply 'CD').

ABCD offers a full ISIS interface in a web environment (using PHP-programming) including all typical ISIS definitions such as 'Field Definition Table', 'Field Selection Table' and the crucial Formatting Language. Thus, it facilitates any structure to be created by the software itself and subsequently being managed (cataloguing, OPAC and circulation). Moreover, though ABCD is fully independent from the bibliographic structures used, it offers some widely used standards such as MARC (21 and UNI) or CEPAL (a bibliographic format derived from UNISIST, widely used in Latin America). That is, libraries, documentation centres, museums, etc can use the software while retaining their own dedicated information structures. Moreover, since the software is fully web-based, it offers a wide range of possibilities to integrate related but different functions into a web based system, offers multi-media capabilities, full electronic document handling, and paving the way for digital libraries. It is special in the sense that it eradicates the need to develop skills (and/or maintenance resources) for running different systems for different purposes with a common 'information service' functionality. The software incorporates all major and important current standards in information services. Moreover, ABCD comes with its own OAI-PMH server allowing harvesting of records by other non-ISIS based web services (de Smet, 2010).

Generally speaking, ABCD represents the culmination and integration of many mostly BIREME-developed ISIS tools of meta-search capability (i.e., including any number of internal and external databases into the search), the CMS-based library 'portal', advanced Serials Management (for local and union catalogues of serial, also electronic, publications), and finally the Advanced Loans module which allows for linking to non-ISIS user databases and multiple loans policy implementations. The ABCD software therefore is best presented as a 'suite' of cooperating but also independent software packages as can be seen from the following figure (de Smet, 2010).



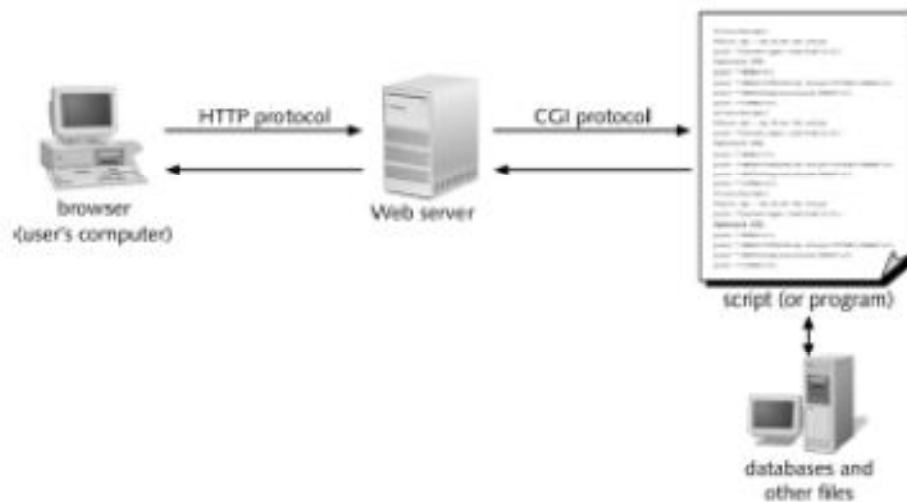
Source: de Smet, 2010

**Figure 2: ABCD Suite**

As can be seen from the above figure, the ‘central’ module contains the ‘heart’ of the software with the main crucial library automation functions, enabling smaller libraries with insufficient technological skills to fully automate without leaving the familiar ISIS environment (as most ABCD users are expected to be current ISIS users) and building upon that often significant experience: locally developed database structures and print formats, or indeed the available skills in using the ISIS Formatting Language, can still be re-used while migration to more professional or more widely used standards is still easily within reach (de Smet, 2010).

Moreover, ABCD uses Common Gateway Interface (GCI) protocol to establish a communication between web server and CGI scripts. The Common Gateway Interface provides a consistent way for data to be passed from the user's request to the application program and back to the user. This means whenever a user sends a request CGI will be invoked by the client side using of GET or POST method. For every request, a new process will be created and the request will be stored in the memory of the operating system of the machine. By means of the CGI protocol, results for users request will send back to the browser. To accomplish this process CGI will call, in the case of ABCD, the command based on mx or wxis executables. Mx is a program for ISIS database which carries out most ISIS

database interface functions and executed from operating system command line to perform different operations (searching, adding, updating and deleting) using different parameters. The other CGI-executable is 'wxis', a web based version of mx with the addition of a built-in scripting language 'IsisScript' to manipulate ISIS-records with XML-style scripts (de Smet, 2009). The general architecture of using CGI protocol is shown below:



Source: Hahmann, 2008

**Figure 3: CGI Protocol Architecture**

### ***3.3. Technical Concepts of ISIS as a Semi-Structured NoSQL Database***

This section is mainly based on materials gathered from experts on ISIS, partly unpublished. In the previous sections we have discussed the history and wide varieties of applications of CDS/ISIS in libraries, documentation centres, and museums. We have seen also how it developed over time following different technological improvements. Moreover, we have seen the ABCD software and its wide application especially in developing countries. In this section, we will briefly present the technical concepts of ISIS.

According to Ramalho (2011), "Generally speaking, ISIS is categorized under the semi-structured databases models. The semi-structured data model is designed as an evolution of the relational data model that allows the representation of data with a flexible structure. Some

items may have missing attributes, others may have extra attributes, some items may have two or more occurrences of the same attribute. The type of an attribute is also flexible: it may be an atomic value or it may be another record or collection. Moreover, collections may be heterogeneous, i.e., they may contain items with different structures. The semi-structured data model is a self-describing data model, in which the data values and the schema components co-exist”.

De Smet (2010) has summarized these technical concepts (the database storage engine, ISO 2709, the ISIS B-Tree indexing, the ISIS-Query language, and the ISIS-Formatting Language) which constitute the common ‘family characteristics’ of the ISIS software family as follows:

1. ‘scheme-less’ records (i.e., no fixed structure is imposed on the records, but each record carries its own ‘identifier’ or ‘fingerprint’ as an ISO 2709 header describing structure and contents) with variable length and structure (i.e., fields of any length can be present 0 up to x times in any record). All data are stored in fields and subfields which are simply identified by their numerical tag followed by the string of the field data; momentarily, this basic ‘tag value’ approach is again gaining interest in computer science thanks to a new ‘NoSQL’ movement (see e.g., NoSQL, 2009; Lai, 2009).
2. All data from the fields go through a ‘filter’-mechanism steered by a detailed and powerful ‘formatting language’. This language acts as a ‘parser’ over the records to retrieve and format any part of the data. This grammar enables librarians to define the use of their data in very detailed and advanced ways (including relational use of data from other databases) without having to become real programmers, and in our view this is the basis of ISIS’s rather unique capability to allow for sophisticated local development and management in situations where computer science skills and knowledge are often very limited.
3. Based on the ISIS Formatting Language values in the databases can be indexed in many ways into an ‘Inverted File’ where all extracted strings are accessible in an almost volume-independent fast (B-tree) structure for immediate retrieval of the

records (with exact information on field occurrence and word positions) linked to that extracted value.

4. The Formatting Language also allows for importing and exporting the data according to very specific instructions, largely taking away the need for conversion expertise when migration is needed.

### 3.3.1. The ISIS Master File (MST)

The ISIS database stores and retrieves information as records. These records specifically are called Master Files Records. ISIS assigns a unique number (id) to each record and this number will be used to identify a record and it is called Master File Number (MFN). Master File Record is a variable length record. In this context, the variable length refers to the number of fields and sub-fields contained by a record and their lengths. The general layout of the Master File Record is based on the ISO 2709 standard for information exchange (UNESCO, 1989).

Header	Data
--------	------

**Figure 4: ISIS Master File Record Structure**

An extension on ISIS Master File has been done which is referred by ISIS Master File Extension implementation (IsismasterFileExImpl), The Isis Master File contains 2 physical disk files which are master disk file and cross-reference disk file. These XRF files are paged disk files (sequence of identical-length pages). The size of each pages ranges from 512 to 65 536 bytes (Dauphin, 2003).

The master or MST disk file is an unordered set of variable-length records which represents the actual content of the records. Each record is identified by combining the value of page\_number and offset. New pages are always appended at the end to insert a new record. Moreover, every time compaction of the files should be done to reclaim storage freed by deleted records (Dauphin, 2003).

As such ISIS implements quite closely the ISO 2709 standard in its Master File, with the exception of the numerical leader/header being binary (for optimized speed) rather than in ASCII as in the standard. The most used and best known implementation of this ISO 2709 standard is the MARC bibliographic standard as designed by the Library of Congress in Washington (see <http://www.lc.gov>). MARC (Machine Readable Cataloguing) is a standard, although coming in several varieties (e.g. MARC21 and UNIMARC) for the representation and communication of bibliographic and related information in machine-readable form, and related documentation. It provides the protocol by which computers exchange, use, and interprets bibliographic information.

In ISO 2709 a record contains four different sections which are record label, directory, data fields and record separator. The record label in ISO 2709 represents the first 24 characters of a record which has fixed length and it includes the record length and the base address of the data contained in the record. The first five positions of the record length represent the size of the record. The maximum number of characters which can be represented by the first five characters is 99999. This means ISO 2709 limits the size of a record to 99999 (100KB). The size limitation of the classical ISIS comes from this fact. The directory section is the second section of ISO 2709; it provides the entry positions to the fields in the record, along with the field tags. A directory entry has four parts which are field tag, length of the field, starting character position of the field and the last optional part is implementation defined part and this part cannot exceed nine characters in length. The data field section, which is the third and main section of ISO 2709, has variable length and it represents the field and subfield data in the record. The last section is the record separator which contains a single character. Moreover, fields in a record are divided in three types such as record identifier field (identify the record), reserved fields (fields reserved for processing of the record) and bibliographic fields which contain data and a field separator (Marbi, 1996).

A major advantage of the ISO 2709 standard is that by parsing only the header, the whole record, which can be of variable structure and length, is 'known' to the software without a need to parse the whole contents also, as is the case with XML.

### **3.3.2. The ISIS Cross-Reference File (XRF)**

The cross-reference disk file is used as a virtual linear one-dimensional array of data objects of fixed size indexed by the master file number which can be randomly accessible. It simulates a large variable-length one dimensional array indexed by the MFN which gives back the data object element which contains the address of the record associated with the MFN on the master disk file, i.e. the pair of values (page\_number, offset) and it can be considered as the ‘primary key’ index on the database file ‘MST’. Without this XRF the MST is like a city without a map: unstructured and inaccessible. However there are tools available to reconstruct this index from the MST when it got lost or corrupted (Dauphin, 2003; BIREME, 2007).

### **3.3.3. The ISIS Inverted File (IF)**

ISIS uses a combination of a B-Tree organized list of values taken from the records (extracted as defined by the ISIS Formatting Language, see *infra*) and an alphabetically list as a ‘dictionary’ in an ‘Inverted File’. This acts as an overall secondary index on the database but with added information: not only the source record (MFN) is noted down in the Inverted File, but also the exact location of the generated value within the MFN: the field tag, the occurrence of that field, and for full-text indexing the position of the word within that occurrence of the field. This allows very fast retrieval of records and combinations into Boolean logics sets by comparing hit-lists with either the AND, OR or NOT logics (UNESCO, 1989).

### **3.3.4. The ISIS Query Language**

”The search language of CDS/ISIS is based on Boolean algebra, which provides a convenient way of expressing logical operations between classes. Each search term associated with a given record, in fact, can be viewed as representing the class of all those records associated with that term” (UNESCO, 2004).

According to the UNESCO (1989), the ISIS Query Language contains the following elements:

- Keywords from the dictionary which has been built from applying the ‘extraction formats’ in the FST onto the database fields
- Boolean operators AND/OR/NOT

- Priority parentheses ( and )
- Right-truncation \$
- Proximity operators (a number of dots indicating the maximum distance in between two words)
- Same-field and same-occurrence of the field operators G and F for use with proximity operators
- Field-limiters (to define in which field the word is required to occur)
- A special operator 'any' allows inclusion of search terms defined in a special file for given words, e.g. to include 'medicine' when searching for 'health sciences'
- Free-text search by starting the search with a question mark (?)

This language represents a typical 'Boolean' search engine. Not all versions of ISIS support the proximity operators, e.g. the extended CISIS (with 1MB max size records) is optimized for much faster indexing of (many) larger documents and dropped including the word-positions into the indexes.

Typically ISIS searches also will run starting from a selection out of the dictionary with search-terms.

### **3.3.5. The ISIS Formatting Language**

The ISIS Formatting Language is a 'parser' of the data in the ISIS-fields to extract values, with additional instructions on how to present the extracted data. These instructions are kept in 'Print Format Tables' or PFT's. Originally designed for non-graphical DOS- and UNIX-environments (with 80-characters fixed position lines), in the Windows version it included many typical Windows-feature, such as font-selection, colours, hyperlinks etc.

The central operator is the 'v'-operator, resulting in the 'v'alue of the field identified with the tag following the v-operator. E.g. v11 will present the contents of field with tag 11.

Around these field values lots of things can be constructed to make the actual application. E.g. logical routing (IF...THEN...ELSE...FI) can be applied as well as other programming elements (like using a local 'counter' in a loop), but mostly ISIS-applications are built around displaying field-values in a variety of flexible formats. Many library- and other applications have been built just on top of this Formatting Language of this general-purpose database

environment, by local librarians without real programming skills or IT-support. This is a very important feature of the ISIS-FL and keeping that means keeping the ‘heart’ or core of ISIS intact, no matter what other technologies are used (as is the case in J-ISIS).

In the current web-based versions no specific provision is made for HTML- or XML-tags since these can be perfectly added using the ‘literals’ (fixed pieces of text) to be added to the field values. For coping with the flexibility of field structures, literals can be inserted with three types of quotes :

- Single quotes ‘ ’ make the literal ‘unconditional’, meaning they will be printed anyway regardless of any condition;
- Double quotes “ ” make the literal behave special for ‘repeatable’ fields : regardless of the number of occurrences of the field, it will be printed only once;
- Pipes || make the literal ‘conditional’, where the condition is that the field attached to it (i.e. mentioned without a comma in between) is present. This way, literals like field names, can be printed only if the value of the field is present.

Another concept of the ISIS-FL from the beginning was the use of 3 different modes:

- Mhl/u : display the values as they would be printed in ‘headings’, i.e. hiding some internal codes such as subfield codes and markers like < > and //
- Mdl/u : display adjusted for data, by adding dots at the end
- Mpl/u : displaying for proof-reading, i.e. all field values without any alteration.

The l/u option defines whether or not to convert into upper-case.

The possibility to ‘hide’ (h-mode) markers was used to allow marking words for indexing. ISIS used the < and > brackets, long before these became the HTML-tag markers. Therefore in web-based systems, where HTML is to be inserted, one has to make sure the H-mode is not used in order to keep the tags recognized as HTML-tags.

In WinISIS and ABCD the production of default formats for just displaying the values of the records is automated, taking away the previously needed skills to create formats by hand. Typically ISIS-formats look quite ‘inaccessible’ for non-experts, but in fact represent some few basic elements repeated many times to deal with all the fields in a database.

In J-ISIS a PFT-editor is available with grammar recognition and validation, which was not available before in any other ISIS-family member. The same approach is now used in the brand-new Alpha-ISIS. Both prove that the FL is indeed the core of the ISIS-technology: they substitute the database-storage layer and indexing for more modern solutions but keep the ISIS-compatibility by keeping the FL, although with always more and new extensions.

One very powerful feature of the FL is to use data from other databases, making ISIS 'run-time' relational. This means that at run-time, when data are actually needed, links can be made in between ISIS-databases (like in a set of tables), using the powerful ISIS-indexing to identify records in databases. This semi-relational feature is fully used e.g. in library systems to link catalogs to loan-objects and users.

Since the FL not only is used to present data to users, but also to other functions (methods) in the system e.g. export-functions, sorting mechanisms and indexing, this relational feature can be used while creating the indexes. As an example: codes in a database could be indexed by their values retrieved from this REF-> (L->()) combined functions. The REF-> function refers to another database (the name of which follows the arrow) and then a PFT will be applied to the record from that other database which is identified by the 'L'lookup function which actually searches the given value quickly in the index and returns the MFN. Finally the ISIS FL contains lots of typical functions, like substrings, average, max and min of numerical ranges etc.

## **Chapter Four: Technical Concepts of the New Generation ISIS Software (J-ISIS)**

### **4.1. Introduction**

In the previous chapter, we have seen the history and evolution of the ISIS software family. We have also discussed the various application of ISIS in relation to a wide range of national and international organizations. In addition, we have discussed about the ABCD integrated library software in relation to ISIS and its wide application particularly in developing countries. Finally, we have also made an emphasis on some of the technical concepts of ISIS. In this chapter, we will discuss the new and latest (fifth generation) ISIS technology, which is Java ISIS (J-ISIS) and its features and technical concepts. At the end of this chapter, a brief comparison of the ISIS and J-ISIS is presented so as to highlight the fundamental improvements of J-ISIS over its previous generation ISIS, specifically regarding techniques for digital libraries.

### **4.2. Java Integrated Set for Information Services (J-ISIS): The Client-Server Java Implementation of ISIS**

An older-generation ISIS for Java (JavaISIS), which was conceived and written by Renate Enea and made available in 1998 by DBA in Italy, should not be mistaken for the new J-ISIS. From its third release onwards it has been distributed and supported by UNESCO, which by then had become a multi-user client-server data entry module with ISO 2709 import and export using WWWISIS as its web engine (Buxton, 2010). Therefore this JavaISIS is a representative still of the ‘classic’ old ISIS technology (see the ISIS software family picture above).

J-ISIS, not to be mistaken for JavaISIS, can be defined as the new ISIS for Java. It is a new multi-platform FOSS ISIS software package which solves the limitations of its previous generation, the CDS/ISIS. J-ISIS is characterized by its sophisticated and flexible Client-Server application, implementation of Unicode, and its usage of a NoSQL database (Berkeley DB) and a new technology for indexing and searching, Lucene, which are based on the latest software developments (Hopkinson, 2008). Moreover, the main objective of developing J-

ISIS was to provide a long-term solution with qualities of modular, easy maintainable and extensible by keeping some of the best qualities of the classical ISIS (Hopkinson, 2008).

Despite all the above advantages and improvements over the previous generations, there are also plans to develop a JDBC (Java Data Base Connectivity) driver and servlets that would allow web-based access to the Database Server by means of an application server such as Tomcat or GlassFish. As the result, the database server will be a scalable NIO (New IO) server, which was introduced in recent versions of Java J2SE 1.4, that use multiplexing and can then accept thousands of clients (Hopkinson, 2008).

J-ISIS is a Client-Server based application which is working as both a database server and a client. When one starts J-ISIS, the J-ISIS database server will be started, listening on port 1111 by default. Different requests and results are passed as messages through TCP/IP. However, as a client, one can connect to either 'localhost' database server of the local machine or any other machine having J-ISIS running and it will also used as J-ISIS database server , for which one has to provide an IP address of the machine as a 'host name'. The J-ISIS database server provides the business data of the application (Dauphin, 2010a).

### **4.3. Web-JISIS**

Web-JISIS is a web-interface which is running on top of the J-ISIS server. Web-JISIS is a Rich Internet Application (RIA) which uses a 'Three-Tier Architecture', which is presented as follows (Dauphin, 2010b):

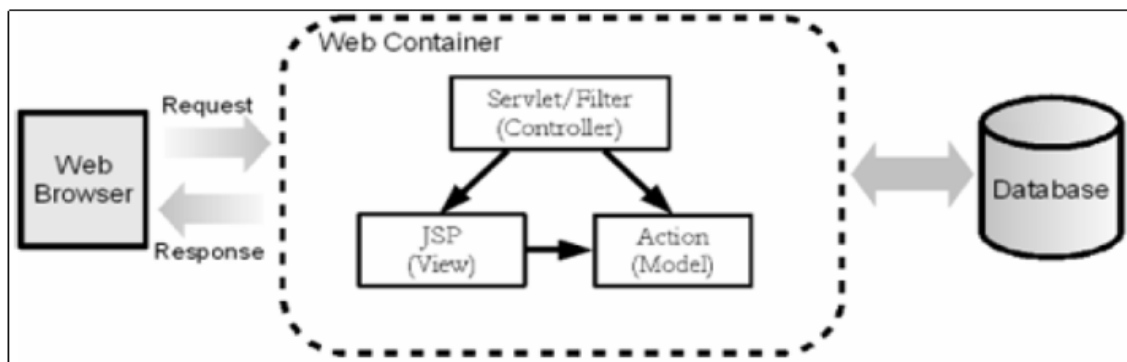
*Tier 1: At the client side, Web 2.0 technologies are responsible for the application's graphical user interface. AJAX is used to communicate between this tier and the application server in tier 2. Requests are sent with the URL using GET or POST data. Request and Result data are formatted using JSON (Java Script Object Notation), XML (Extensible Markup Language), or plaintext.*

*Tier 2: the middle tier is also known as the application server, which provides the business processes logic and the data access. The server in the middle tier is a servlet container—a Web server capable of running Java-based Web applications (Tomcat, Jetty, etc). The J-ISIS services (the business rules) are coded in this tier using Java and the client part of jisiscore.jar library. Requests are passed as messages to the database server and results are*

returned. Sockets are used to communicate between this tier with the client and the database server through TCP/IP. Specifically this tier based on Struts 2 framework. Struts 2 is a framework for developing Model 2 applications. It makes development more rapid because it solves many common problems in web application development by providing the following features: page navigation management, user input validation, consistent layout, extensibility, internationalization and localization and Support for AJAX.

Servlet technology and Java Server Pages (JSP) are the main technologies for developing Java web applications. When introduced by Sun Microsystems in 1996, Servlet technology was considered superior to the reigning Common Gateway Interface (CGI) because servlets stay in memory after responding to the first requests. Subsequent requests for the same servlet do not require re-instantiation of the servlet's class, thus enabling better response time than of CGI technologies.

The recommended architecture for Java web applications today is called “Model 2”. Model 2 is another name for the Model-View-Controller (MVC) design pattern. An application implementing the MVC pattern consists of three modules: model, view, and controller. The view takes care of the display of the application (JSP). The model (Action) encapsulates the application data and business logic. The controller (Servlet/Filter) receives user input and commands the model and/or the view to change accordingly.



(Source: From Web-J-ISIS manual)

**Figure 5: Web-JISIS Three Tier Architecture**

Tier 3: the J-ISIS database server provides the business data. The J-ISIS database server is listening for request on port 1111, requests and results are passed as messages through TCP/IP.

Since Web-JISIS is based on three-tier architecture, it benefits from the advantages of three-tier architecture applications, specifically, it is easier to modify or replace any tier without affecting the other tiers; separating the application and database functionality means better load balancing; and adequate security policies can be enforced within the server tiers without hindering the clients.

#### **4.4. NoSQL Database Technologies**

NoSQL, 'Not SQL' or alternatively "Not only SQL" (both versions are used) is a non-relational database technology which is designed to execute a very large volume of simple updates which supports millions and potentially even hundreds of millions of online users and reads against a single, very large dataset (Oracle, 2011). This doesn't mean NoSQL is replacing the traditional relational database technology; rather, both technologies are coexisting and can be used based on the type of application.

"Many NoSQL databases trade off "ACID" (Atomicity, Consistency, Isolation and Durability) guarantees in favour of providing for very-high performance in the broad scale/simple store and retrieve scenario" (Brust, 2011).

According to Oracle (2011), unlike RDBMS, in case of NoSQL databases, the process of supporting hundreds of online users for updating and reading from a very large dataset is accomplished by horizontally scaling across large number of servers. This will distribute the application load into several servers. As a result, changes to the application can be made without letting the application down. For instance, application behaviour can be changed by replacing the software in individual server (CouchBase, 2011).

The relational database technology is invented in the 1970s and is still widely used as it was optimized for the applications, users and infrastructure of that era. However, despite the fact that the number of 'band aids' have extended the useful life of the technology (such as horizontal and vertical sharding , distributed caching and data denormalization), the need for modern interactive software systems has increased due the fact that the cost of the total system and its complexity has increased over time (Couchbase, 2011).

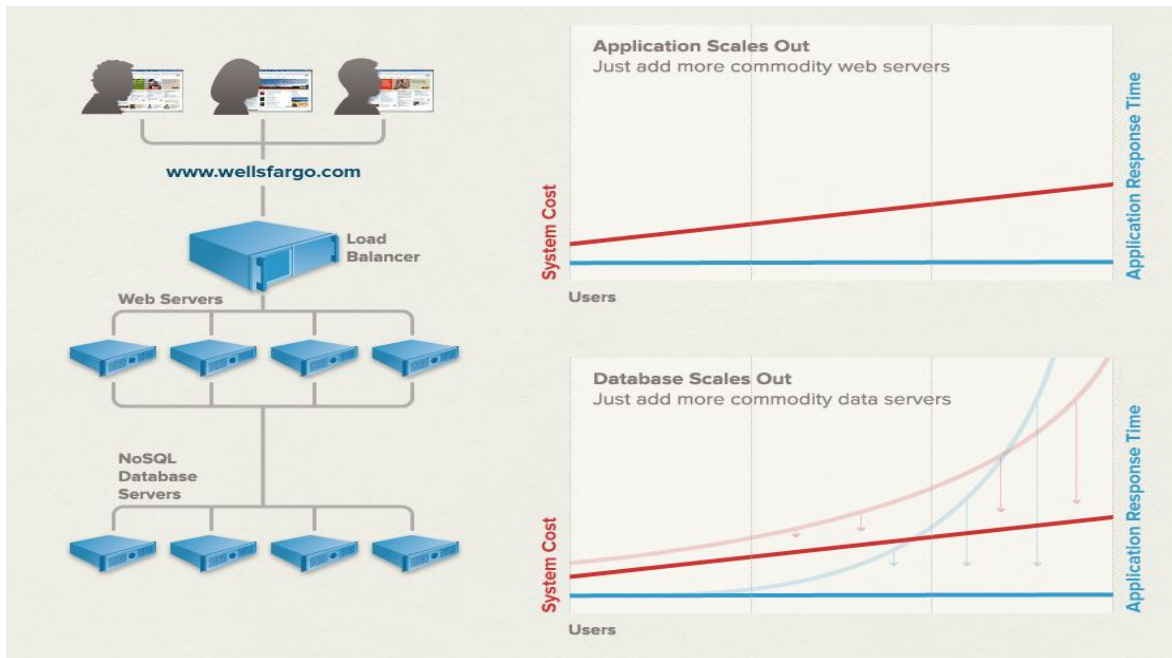
Accordingly, organizations such as Google and Amazon were forced to invent “NoSQL” or non-relational database technologies for better scalability. Over the last 40 years, there have been changes and fundamental differences in the users, applications and underlying infrastructure of the software systems. The application architecture has also been changed. That is, in contrast to the scale-up, centralized approach of circa 1975 interactive software architecture, modern Web applications are built to scale out (simply add more commodity Web servers behind a load balancer to support more users). However, relational database management systems (RDBMS) technology, which is a “scale-up” technology, has not fundamentally changed in the last 40 years and hence continued to be the default choice for holding data behind web applications. The RDBMS technology has a lot of limitations when applying to modern interactive software systems due to its rigidity of the database schema (i.e. schema should be strictly defined before storing any data into the database). Thus, in the modern world, where there is rapidly evolving business and market environment, the use of RDBMS technology is not anymore always suitable.

Couchbase (2011) has summarized the characteristics of NoSQL database management systems as follows:

1. *No schema required. Data can be inserted in a NoSQL database without first defining a rigid database schema. As a corollary, the format of the data being inserted can be changed at any time, without application disruption. This provides immense application flexibility, which ultimately delivers substantial business flexibility.*
2. *Auto-sharding (sometimes called “elasticity”). A NoSQL database automatically spreads data across servers, without requiring applications to participate. Servers can be added or removed from the data layer without application downtime, with data (and I/O) automatically spread across the servers. Most NoSQL databases also support data replication, storing multiple copies of data across the cluster, and even across data centres, to ensure high availability and support disaster recovery. A properly managed NoSQL database system should never need to be taken offline, for any reason, supporting 24x7x365 continuous operation of applications.*

3. *Distributed query support. “sharding” an RDBMS can reduce, or eliminate in certain cases, the ability to perform complex data queries. NoSQL database systems retain their full query expressive power even when distributed across hundreds or thousands of servers.*
4. *Integrated caching. To reduce latency and increase sustained data throughput, advanced NoSQL database technologies transparently cache data in system memory. This behaviour is transparent to the application developer and the operations team, in contrast to RDBMS technology where a caching tier is usually a separate infrastructure tier that must be developed to, deployed on separate servers, and explicitly managed by the ops team.*

As can be seen from the following figure, the use of NoSQL database technology flattens the non-linear increase in total system cost curves and asymptotic degradation of performance (application response time) with RDBMS technology.



Source: Couchbase (2011)

**Figure 6: How use of NoSQL Flattens the Total System Cost and the Asymptotic Degradation of the Performance**

At this time, there are a fairly large number of NoSQL databases which are being developed, such as Apache Cassandra, MongoDB, Voldemort, Apache HBase, SimpleDB BigTable (the Google database technology), which are in their early stages of development and most of the early adopters employ programming staffs that participate in their development (Oracle, 2011).

NoSQL databases don't support 'joins', transactions that involve updates to data in multiple tables or even multiple rows within a table, and online schema changes (many are even "schema-less") as they are complicated and expensive. Thus, NoSQL databases provide partial functionality in some of these areas by supporting some mechanism for adding new data on the fly. Although there are variations in the exact functionality that the many different NoSQL databases provide (including document, columnar, and key/value approaches), the core idea is that there is some way to add new data elements or columns to tables and the values contained in those columns in a manner that applications can readily detect when the new columns are there or not. One of such supports is the 'Key/value' support which is flexible and straightforward to add additional logic into applications to deal with online schema changes. Every data value is accompanied by a key, which labels the data much like a column name. Applications can be programmed to test and appropriately deal with whether a new column and its value exist yet or not on a row-by-row basis. Key/values also provide partial functionality so as to deal with the lack of joins by allowing de-normalized or pre-joined data to be stored in a single data set (or table) (Oracle, 2011).

To sum up, NoSQL was invented to provide a descriptor for a variety of database technologies which are emerged to provide the "Web-scale" or "Internet-scale" demands. It provides advantages of big data management; huge number of users; and complex data of commercial and business applications (Tweed and James, 2010). Both NoSQL and other database technologies should co-exist as they have advantages for different purposes with respect to different applications.

## 4.5. Berkeley DB

J-ISIS uses Berkeley DB to overcome the size limitation of records which was the main concern of switching from ISIS to J-ISIS. Berkeley DB is an Open Source embedded database system with a number of key advantages over comparable systems. It is simple to use, supports concurrent access by multiple users, and provides industrial-strength transaction support including surviving system and disk crashes. It can be used in applications that require high-performance concurrent storage and retrieval of key/value pairs. The software was originally distributed by Sleepycat as an Open Source product and provides a variety of programmatic interfaces, such as callable APIs for C, C++, Java, Perl, Python, and Tcl (Olson et al., 1999). Now it is property of Oracle but keeps its free and open source nature.

Berkeley DB began as a new implementation of a hash access method to replace both Bsearch and the various dbm implementations (dbm from AT&T, ndbm from Berkeley, and gdbm from the GNU project). Olsen et al. (1999) summarized the history of Berkeley DB as follows:

1. *In 1990 Seltzer and Yigit produced a package called Hash to replace both hsearch and the various dbm implementations.*
2. *The first general release of Berkeley DB, in 1991, included some interface changes and a new B+ tree access method.*
3. *At roughly the same time, Seltzer and Olson developed a prototype transaction system based on Berkeley DB, called LIBTP [Selt92], but never released the code.*
4. *The 4.4BSD UNIX release included Berkeley DB 1.85 in 1992. Seltzer and Bostic maintained the code in the early 1990s in Berkeley and in Massachusetts. Many users adopted the code during this period.*
5. *By mid-1996, users wanted commercial support for the software. In response, Bostic and Seltzer formed Sleepycat Software. The company enhances, distributes, and supports Berkeley DB and supporting software and documentation.*
6. *Sleepycat released version 2.1 of Berkeley DB in mid-1997 with important new features, including support for concurrent access to databases. The company made about three commercial releases a year, and most recently shipped version 3.1.*

#### 4.5.1. Berkeley DB Product Family Overview

The Berkeley DB product family consists of three products: Berkeley DB, Berkeley DB Java Edition and Berkeley DB XML. These products are available as libraries with simple, proprietary APIs for data access and manipulation as well as database administration (see: <http://www.oracle.com/technetwork/database/berkeleydb/overview/index.html>).

➤ **Berkeley DB:**

Although Berkeley DB has been used as the storage engine for SQL database products, it does not support SQL. A typical Berkeley DB application makes API calls to start and end transactions, store and retrieve data as well as to perform administrative functions such as checkpoints and backups. Therefore, a Berkeley DB application is completely "self-contained" with respect to all data management activities which enables a zero manual administration approach to application development and hence it is crucial. This capability in a large number of applications including embedded applications, where manual administration is impossible.

➤ **Berkeley DB Java Edition:**

This is a 100% pure Java implementation which has similar APIs and capabilities for data management with Berkeley DB.

➤ **Berkeley DB XML:**

This is fully implemented in C++ and it is an XML database engine with XQuery and XPath capabilities. It layers on top of Berkeley DB and uses it for storage, indexing, transactions and other database capabilities.

#### 4.5.2. Comparison among the Berkeley DB Family Products

Berkeley DB and Berkeley DB Java Edition are very similar with respect to the features and functionality they provide, though they are architecturally quite different. Architecturally, Berkeley DB is similar to the "update-in-place" architecture of most other traditional database systems. Berkeley DB Java Edition, on the other hand, uses log-structured storage for managing on-disk data. Every change results in a new entry in the log. A separate garbage collector thread that runs in the background reclaims space occupied by obsolete data. Thus, though there are commonalities between the test suites for Berkeley DB and Berkeley DB Java Edition with respect to testing API behaviour, the Berkeley DB Java Edition test suite

also contains specific tests for exercising the log cleaner, "out of disk space" scenarios, log archiving and other aspects specific to the log structured architecture of Berkeley DB Java Edition.

1. *Berkeley DB is implemented in C whereas Berkeley DB Java edition is purely java implementation.*
2. *Both products have similar APIs and capabilities for data management.*
3. *From a testing point of view, porting is not as big an issue for Berkeley DB Java Edition, since the JVM is inherently portable.*
4. *Both products provide indexed access to data; Berkeley DB supports B-trees as well as hash indexing whereas Berkeley DB Java Edition only supports B-tree indices.*
5. *Both products support concurrent access to data. Berkeley DB permits concurrent threads, or concurrent processes or both, whereas Berkeley DB Java Edition typically supports multiple threads within a process and more limited multi-process access.*
6. *Both products support transactions, including support for the various ANSI isolation levels. A row is simply an opaque key:value pair; Berkeley DB does not have the notion of data types, but the Direct Persistence Layer of Berkeley DB Java Edition does provide an optional schema-like capability. Interpreting the opaque contents of the row is left entirely up to the application.*
7. *APIs for administrative operations like database checkpoints and backups are provided by all three products.*
8. *Berkeley DB XML manages XML documents which can either be stored as whole documents, or as individual nodes.*
9. *Berkeley DB XML creates indices on various attributes to improve access performance. Since Berkeley DB XML is layered on the Berkeley DB database engine, it can leverage the test suite of the underlying storage engine, including the replication and high availability features.*
10. *Berkeley DB XML also leverages the XML standards specifications in order to test the correctness of XML processing.*

#### 4.5.3. How Berkeley DB is Used

The Berkeley DB library supports concurrent access to databases. It can be linked into standalone applications, into a collection of cooperating applications, or into servers that handle requests and do database operations on behalf of clients. It is easy to understand and simple to use as compared to using a standalone database management systems. The software stores and retrieves records, which consist of key/value pairs, which are used to locate items and can be any data type or structure supported by the programming language. In J-ISIS is done by using either ISO 2709 or XML in that value-part.

The programmer can provide the functions that Berkeley DB uses to operate on keys such as a custom comparison function for B<sup>+</sup> trees, a custom hash function for Hash access method. In addition, Berkeley DB uses default functions if none are supplied; Berkeley DB does not examine or interpret either keys or values in any way since they may be arbitrarily long (Olsen et al., 1999).

Berkeley DB is neither a database server that handles network requests nor is an SQL engine that executes queries. Moreover, it is not a relational or object-oriented database management system and has been designed to be portable, small, fast, and reliable. A single database managed by Berkeley DB can be up to 2<sup>48</sup> bytes, or 256 peta bytes, in size. Berkeley DB uses the host file system as the backing store for the database, so large databases require big file support from the operating system. Sleepycat Software has customers using Berkeley DB to manage single databases in excess of 100 gigabytes (Olsen et al., 1999).

Olsen et al. (1999) have summarized the main features of Berkeley DB as follows:

1. **Programmatic interfaces:** Berkeley DB defines a simple API for database management. The package does not include industry-standard programmatic interfaces such as Open Database Connectivity (ODBC), Object Linking and Embedding for Databases (OleDB), or Structured Query Language (SQL). These interfaces, while useful, were designed to promote interoperability of database systems, and not simplicity or performance.
2. **Working with records:** Berkeley DB supports both keyed access, to find one or more records with a given key, and sequential access, to retrieve all the records in the database one at a time. The order of the records returned during sequential scans depends on the access method. B<sup>+</sup> tree and Recno databases return records in sort order, and Hash

databases return them in apparently random order. Similarly, Berkeley DB defines simple interfaces for inserting, updating, and deleting records in a database.

3. **Long keys and values:** Berkeley DB manages keys and values as large as  $2^{32}$  bytes. Since the time required to copy a record is proportional to its size, Berkeley DB includes interfaces that operate on partial records. Berkeley DB allows the programmer to define the data types of keys and values. Developers use any type expressible in the programming language.
4. **Main memory databases:** Berkeley DB is able to manage very large shared memory regions for cached data pages, log records, and lock management. Berkeley DB can memory-map its database files for read-only database use. The application operates on records stored directly on the pages, with no cache management overhead. Because the application gets pointers directly into the Berkeley DB pages, writes cannot be permitted. Otherwise, changes could bypass the locking and logging systems, and software errors could corrupt the database. Read-only applications can use Berkeley DB's memory-mapped file service to improve performance on most architecture.
5. **Configurable page size:** Although Berkeley DB provides reasonable defaults, developers may override them to control system performance. Small pages reduce the number of records that fit on a single page. Fewer records on a page means that fewer records are locked when the page is locked, improving concurrency.
6. **Small footprint:** Berkeley DB is a compact system. The full package, including all access methods, recoverability, and transaction support is roughly 175K of text space on common architectures.
7. **Cursors:** Berkeley DB includes cursor interfaces for all access methods. This permits, for example, users to traverse a  $B^+$  tree and view records in order.
8. **Joins:** In database terminology, a join is an operation that spans multiple separate tables (or in the case of Berkeley DB, multiple separate DB files). Berkeley DB includes interfaces for joining two or more tables.
9. **Transactions:** transactions must be atomic, consistent, isolatable, and durable. This combination of properties is referred to as ACIDity in the literature. Berkeley DB, like most database systems, provides ACIDity using a collection of core services. Programmers can choose to use Berkeley DB's transaction services for applications that need them. Some of the issues in relation to transactions include:

- **Write-ahead logging:** At any time during the transaction, the application can *commit*, making the changes permanent, or *roll back*, cancelling all changes and restoring the database to its pre-transaction state. If the application rolls back the transaction, then the log holds the state of all changed pages prior to the transaction, and Berkeley DB simply restores that state. If the application commits the transaction, Berkeley DB writes the log records to disk. In memory copies of the data pages already reflect the changes, and will be flushed as necessary during normal processing. Since log writes are sequential, but data page writes are random, this improves performance.
- **Crashes and recovery:** Berkeley DB can survive application crashes, system crashes, and even catastrophic failures like the loss of a hard disk, without losing any data. That is, different system failures can destroy memory, the log disk, or the data disk. Berkeley DB is able to survive the loss of any one of these repositories without losing any committed transactions.
- **Checkpoints:** Berkeley DB includes a check pointing service that interacts with the recovery system. When an application makes a *checkpoint*, all committed changes in the log up to that point are guaranteed to be present on the data disk, too. Check pointing is moderately expensive during normal processing, but limits the time spent recovering from crashes.
- **Two-phase locking:** Berkeley DB provides a service known as two-phase locking. In order to reduce the likelihood of deadlocks and to guarantee ACID properties, database systems manage locks in two phases. First, during the operation of a transaction, they acquire locks, but never release them. Second, at the end of the transaction, they release locks, but never acquire them.

10. **Concurrency:** Good performance under concurrent operation is a critical design point for Berkeley DB. Although Berkeley DB is itself not multi-threaded, it is thread-safe, and runs well in threaded applications. Concurrent database applications may start up a new process for every single user, may create a single server which spawns a new thread for every client request, or may choose any policy in between. Berkeley DB has been carefully designed to minimize contention and maximize concurrency. The cache manager allows all threads or processes to benefit from I/O done by one. Shared resources must sometimes be locked for exclusive access by one thread of control.

To sum up, Berkeley DB offers a unique collection of features, targeted squarely at software developers who need simple, reliable database management services in their applications. Moreover, good design and implementation, and careful engineering are some of the special features which make the software better than many other systems. Berkeley DB is an Open Source product, available at [www.sleepycat.com](http://www.sleepycat.com) for download (Olsen et al., 1999). Berkeley DB nowadays, after Oracle took it over from Sleepycat, is available from the Oracle website ([www.oracle.com](http://www.oracle.com)).

#### **4.6. Lucene**

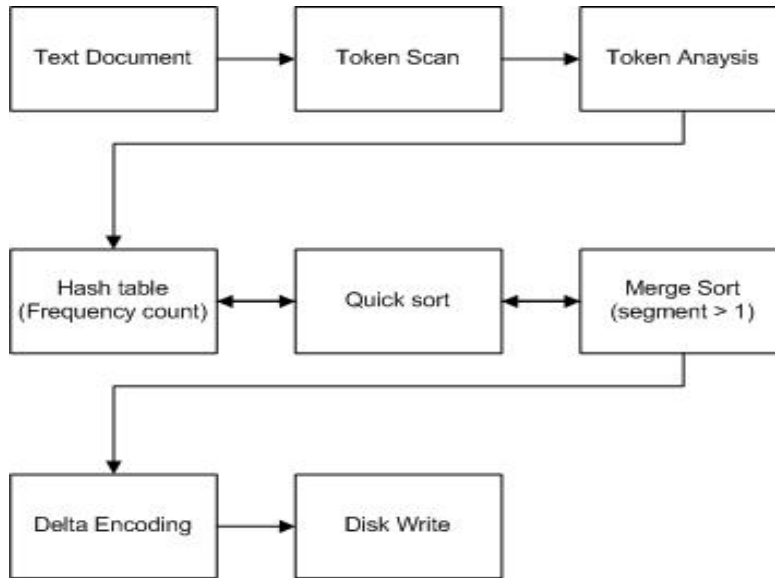
Apache Lucene is a Java based search engine library, which is an open source and freely available and can be downloaded from the apache foundation software website: <http://www.apache.org/dyn/closer.cgi/lucene/java/> and can be integrated into different application by using Java method calls. As a result, it can be used by any application where they want to support a full-text search in their application (Hatcher, 2004). On the other hand, Lucene is not a standalone search application which could be used by its own. Due to this fact, Lucene should be integrated with different applications to facilitate indexing and searching of documents (Prasad & Patel, 2005).

J-ISIS uses Lucene for indexing and searching of documents. This is used for full-text indexing and to enhance the capability of ISIS by adding new features such as giving ranked results and avoiding search key limit of ISIS which was only 60 characters (de Smet, 2010).

Indexing and searching are the two main functions supported by Lucene. Lucene index is a sequence of documents. A document refers to anything written in text and it does not represent actual file in a disk. It is just a record which contains a collection of fields which has a name and the value of the field (Carpenter, 2011). Lucene passes through different steps to index a document. Indexing of a document in Lucene is done by encapsulating the original document, chunking the buffered stream into tokens (tokenizing process), parsing to deal with stop words and Porter stemming, inverting the documents, recording the frequency counts, sorting the postings, encoding the term positions, merging the sub-results if necessary, and storing them onto the disk.

Lucene searching functionality does the process of parsing a query entered by an end-user, performing some versions of optimized search which will make searching faster and finally it performs the reading of relevant data to the user query from disk and returning the results (Su, 2002).

The following figure summarizes the process of indexing in Lucene:



(Source: David Chi-Chuan Su, 2002)

**Figure 7: Summary of Indexing Process in Lucene**

The above figure clearly shows that, Lucene takes the text content of a document as input and it will tokenize the raw text data indexing which means it will divide the text into tokens. In addition, Lucene provides a mechanism for filtering tokens by means of different analyzers. There are different types of analyzers such as white space analyzer (which separates tokens on white space), stop words analyzer (which avoids stop words), simple analyzer (which simply separate tokens on a non-character boundaries, e.g. special characters), standard analyzer (which does the lowercase conversion and removal of stop words, e.t.c. (Hatcher, 2004). After analyzing the tokens, Lucene stores the tokens in hash table for counting the frequency of terms and this could be used later on for ranking of query results. It also uses quick sort and merge sort to sort the posting table and index segments to represent the full

document respectively (Su, 2002). Finally, delta encoding will be implemented to encode posting lists to save space and the index will be written into a disk.

To sum up my discussion about Lucene, let me present the following points about the capabilities of Lucene as summarized by Apache (2011).

- *It is scalable, High-Performance Indexing*
- *100 %-pure java*
- *Common Logging framework which is used for logging of system messages. This can be configured in the settings for critical and non-critical errors and messages*
- *Powerful, accurate and efficient search algorithms*
- *Many powerful query types: phrase queries, wildcard queries, proximity queries, range queries wildcard*
- *Date-range searching and sorting by any field*
- *Multiple-index searching with merged results allows simultaneous update and searching*
- *Desirable Memory Footprint*
- *Incremental indexing as fast as batch indexing*
- *Ranked searching, this refers to best results returned first, in addition to many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more*
- *Fielded searching (e.g., title, author, contents) date-range searching sorting by any field multiple-index searching with merged results allows simultaneous update and searching*

Lucene is widely used nowadays and it is being used by Wikipedia, Technorati, Monster.com, Nabble, TheServerSide, Akamai, SourceForge (Yonik Seeley, 2007).

#### **4.7. Comparison between ISIS and J-ISIS**

Fundamentally, J-ISIS is the successor of CDS/ISIS for Windows in many respects. These include:

- 1) It is a general-purpose interface to manage ISIS-databases, and fully graphical (windows, mouse, and pull-down menus);

2) It implements all CDS/ISIS for Windows Formatting Language features and other functions (GMOD, unlocking, import/export etc.);

3) It presents itself as a single interface without a need for a web-browser or web-server and hence is relatively simple to install and maintain (Hopkinson, 2008). The fact that J-ISIS maintains ISO 2709 as a common data exchange format and enriched with XML as a storage and exchange format enables librarians themselves to use this technology without the assistance of IT-experts, also remains directly compatible, e.g. with REF and L functions to relate to other databases, and other advanced features of CDS/ISIS for Windows (de Smet, 2008; Hopkinson, 2008).

However, there are significant differences between ISIS and J-ISIS. These include:

1) it is Java-based and hence can run on all computer platforms (including Linux, OSX) without a need to convert the databases to the proper platform as is the case with classic ISIS;

2) it is client-server, which allows access to remote (Internet-based) databases in addition to local databases;

3) instead of the classic MST-XRF, it uses Berkeley DB technology which will overcome the limits on current record and database-size and hence allow digital libraries to keep long full-text documents;

4) it will use Lucene indexing instead of the classic ISIS Inverted File, allowing ranked search results on top of Boolean and full-text searching, but based on the classic ISIS FST's (which in turn are based on the ISIS Formatting Language);

5) Existing applications will be compatible with new technologies with minimal effort (export-import);

6) Unicode has been adopted; and

7) ISIS is no longer a 'database' but a standard for handling semi-structured textual databases (Hopkinson, 2008; ISIS3WC, 2010).

In addition, at the occasion of the ISIS3WC conference in 2010, de Smet has summarized the comparison of basic technologies of ISIS and J-ISIS as follows:

**Table 1: ISIS Family Comparison**

Basic technologies : Comparison			
Main characteristics	Classic ISIS	J-ISIS	ISIS-NBP
Programming language	Pascal, C, C++	Java	Python
Database technology	MST + XRF	Berkeley DB	CouchDB
Unicode support	No	Yes	Yes
ISIS FL support	PFT, FST, FMT	PFT, FST, FMT	PFT, FST, FMT
Standard for information interchange	ISO 2709	ISO 2709	ISO 2709
Indexing technique	Inverted file indexing, only Boolean	Lucene indexing, ranking	Lucene indexing, ranking

Source: ISIS3WC, 2010

## Chapter Five: Implementing and Testing Digital Library Technology into J-ISIS

In this chapter, I will present and discuss my practical work and results on the implementation of digital library technology into J-ISIS. This has three parts mainly of testing the existing features and adding digital library feature into the software.

The first part of this chapter (section 5.1.1 to 5.1.4) is devoted to test the already available features in J-ISIS but needed to be tested, such as web-based retrieval of resources, diversity of metadata formats, full-text indexing and Unicode compatibility. Thus, the objective of this part is to show readers practical tests and ensure the practicability of these already available features which will be a nice guide before proceeding to the new digital library function which are added in this paper.

The second part is the central objective of this paper. In this part, I will present the new features which have been implemented for enhancing the capability of J-ISIS specifically regarding the digital library technology and also, in the third part, the analysis of J-ISIS performance in terms of size and indexing speed will be discussed.

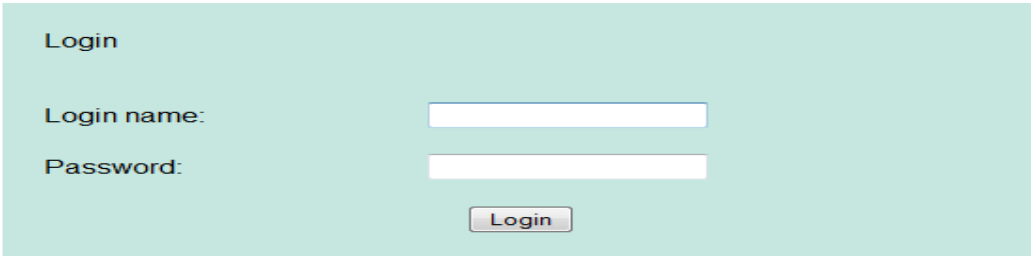
### **5.1. Testing Available Digital Library Features in J-ISIS**

#### **5.1.1. WWW Based Retrieval: Web-JISIS**

J-ISIS has a Java based web application prototype i.e. Web-JISIS, which enables the end-user to use a web-browser to open a J-ISIS database, search resources, browse resources and (in a future version) to edit resources by sending requests using GET and POST methods.

Web-JISIS enhances the flexibility of J-ISIS and it makes J-ISIS a web-based application. Thus, the software can benefit from the advantages of web technologies. When the end-user starts the application server on a local machine (<http://localhost:8080/Web-JISIS3/>) the following login page of Web-JISIS will be launched for the user to login:

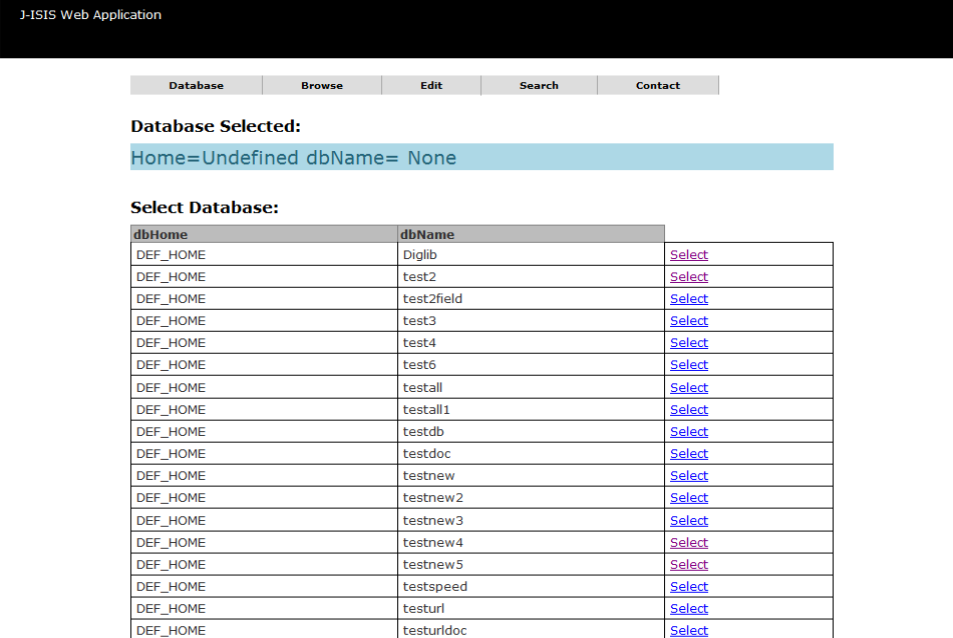
## J-ISIS Web Application



The login form is displayed on a light green background. It features the title "Login" at the top left. Below the title, there are two input fields: "Login name:" and "Password:". Each input field is a simple white rectangle with a thin border. Below the "Password:" field, there is a "Login" button with a grey gradient and rounded corners.

**Figure 8: Web-JISIS Login Form**

Once the user is logged in, requests are sent to the database server via TCP/IP. To get access to the database server from the client side, the database server (from the J-ISIS Suite) should also be running, which is listening to the port number 1111. After the user logged in successfully, the J-ISIS Web application database selection page will be displayed as follows:



The screenshot shows the "J-ISIS Web Application" interface. At the top, there is a navigation bar with buttons for "Database", "Browse", "Edit", "Search", and "Contact". Below the navigation bar, the text "Database Selected:" is followed by "Home=Undefined dbName= None". Underneath, the text "Select Database:" is followed by a table listing various databases. Each row in the table has three columns: "dbHome", "dbName", and "Select". The "Select" column contains a blue "Select" link.

dbHome	dbName	Select
DEF_HOME	Diglib	<a href="#">Select</a>
DEF_HOME	test2	<a href="#">Select</a>
DEF_HOME	test2field	<a href="#">Select</a>
DEF_HOME	test3	<a href="#">Select</a>
DEF_HOME	test4	<a href="#">Select</a>
DEF_HOME	test6	<a href="#">Select</a>
DEF_HOME	testall	<a href="#">Select</a>
DEF_HOME	testall1	<a href="#">Select</a>
DEF_HOME	testdb	<a href="#">Select</a>
DEF_HOME	testdoc	<a href="#">Select</a>
DEF_HOME	testnew	<a href="#">Select</a>
DEF_HOME	testnew2	<a href="#">Select</a>
DEF_HOME	testnew3	<a href="#">Select</a>
DEF_HOME	testnew4	<a href="#">Select</a>
DEF_HOME	testnew5	<a href="#">Select</a>
DEF_HOME	testspeed	<a href="#">Select</a>
DEF_HOME	testurl	<a href="#">Select</a>
DEF_HOME	testurldoc	<a href="#">Select</a>

**Figure 9: List of Database Displayed from Web-JISIS**

As we can see from the above figure, Web-JISIS provides access to the databases stored in J-ISIS as a result of transferring a client request via TCP/IP protocol.

The following figure illustrates browsing of database record from Web-JISIS with a navigation bar on top of the interface and the record display in the page.

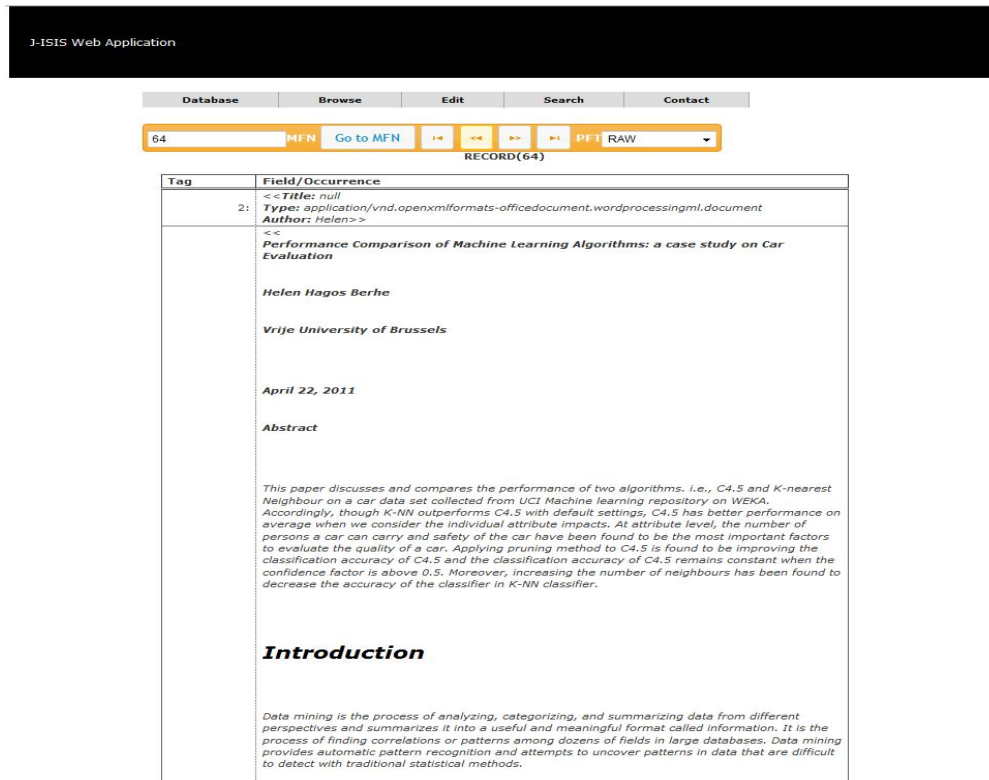


Figure 10: Database Record Display from Web-JISIS

### 5.1.2. Diversity of Metadata Formats

J-ISIS, like any other ISIS software, supports the idea of openness against metadata formats. In technical terms, this means that it allows creation of databases with different or variable number/nature of the fields in a record. This flexibility regarding record structures, which is the essence of 'documentary databases' as opposed to 'administrative databases', makes J-ISIS to handle variable length records within the same database, depending on the necessity of the application related to metadata. For instance, one record can contain three fields to describe a webpage, the second record can contain seven fields to describe a staff-member and the third record in the database can contain five fields for a bibliographic description, e.t.c. In reality bibliographic structures can contain hundreds of fields as e.g is the case in UNIMARC.

This feature, based on the fact that each record itself carries its own structural description (in the header) avoids wastage of space which is a common case in relational databases as the data in the same database should share the same structure, even though the value is "null" . This implies, in case of J-ISIS, that the metadata definition is very flexible whereas the metadata definition for relational databases is more static as we have to define it in advance. Therefore, the librarian will not be forced to use the same number of fields for all records like in relational databases. Splitting up in several tables to avoid this problem is also not required, ending in a much more simple architecture of the data. In cases where no specific advantages of relational databases, like consistency and avoiding repetition of data are required (or not even desired), this allows a more efficient approach in speed and efficiency, e.g. records can be read from the hard disk in one movement of the hard disk head-reader. Moreover, since all data fit into one single 'table', a much simpler setup is possible, also allowing the interface to offer database-creation possibilities without all the hassle of correctly relating many tables, which is mostly for experts only.

In view of the fact that the ISIS database technology is an old technology, which was introduced even before the idea of relational database and schema-less databases became involved , J-ISIS like its other ISIS-siblings, is best called a 'semi-structured database' software (Ramalho, 2011). This means, like any databases it provides the possibility of a table with the definition of possible fields that are expected to be used in the application. But, later on, the librarian can decide to use only part of the fields or even other fields than defined originally, depending on the interest of the librarian and usage of these fields. The interesting part is even if the librarian decides to use part of the field definition table there will not be a wastage of resources such as space, as the database record will be structured by the actual usage of the available fields as described in its own header. This promotes the idea of a "schema-less" database. This is the reason why we call J-ISIS is a semi-structured database software. This fact was proven by doing some tests on using different fields with different records within the same database.

### **5.1.3. Full-text Indexing**

Library software should make its resources searchable and accessible by the end-user. To make library resources accessible, the library software should provide an efficient indexing

and searching method. This implies that library software will be more usable and accessible by end-users.

Indexing of documents using Lucene is incredibly fast, as we found out during our tests. For example: to index a collection of 200 documents it took only 9 seconds on average. As a result, this enhances the capability of J-ISIS as digital library software.

In addition to the fact that Lucene provides full-text indexing, the idea of full-text indexing was already there with the original ISIS and J-ISIS also inherits this characteristic from its predecessors. To benefit from both full-text and 'field-structured' retrieval, ISIS and Lucene have been integrated for defining different level of searching based on the type of particular application. If it was only full-text indexing, only one level could be used which means each words can be used for searching documents whereas by integrating these techniques, different searching levels can be used depending on the application type.

Accordingly, a librarian can define the level or technique of indexing documents for the particular application. The possible types of indexing techniques that could be used in J-ISIS are listed below:

- Indexing technique 0 : lines extracted from the Print Format Table (PFT)  
(used for full-field indexing)
- Indexing technique 1: Subfields & lines
- Indexing technique 2: Terms or phrases <...>
- Indexing technique 3: Terms or phrases /.../
- Indexing technique 4: Each word in extracted text (i.e. full-text indexing)
- Indexing technique 5: Prefixed terms tech 1
- Indexing technique 6: Prefixed terms tech 2
- Indexing technique 7: Prefixed terms tech 3
- Indexing technique 8: Prefixed terms tech 4

This way, by defining the indexing technique for each field of a record, the librarian could define the way of searching in a detailed way. For instance, a digital library application might be interested to use the full-text searching which is defined in the Field Selection Table (FST) as techniques 4 and 8, whereas for bibliographic based libraries might be interested on using other indexing techniques (e.g. indexing technique 1) because they use the metadata for

searching and for this type of searching J-ISIS provides different indexing techniques. A mix of both indexing types (full-field and full-text) can be used, even on the same field. Needless to say that, if the field is a full document text, techniques based on the full-field approach won't make sense.

### **5.1.3.1. Integration of Lucene with J-ISIS**

In the following sections, we will discuss how Lucene has been integrated and used for indexing and searching in J-ISIS. This makes J-ISIS to be fast in indexing and searching of documents, to avoid the 60 character search key limit of classic ISIS and ranking of search results.

#### **Lucene Index Creation**

In J-ISIS, the indexer class in the “org.unesco.jisis.corelib.index” package is responsible for creating, update, and delete an index. For creating the index, we have to specify the location in the file system where the index will be created and J-ISIS uses a server directory called “index”. Lucene uses the J-ISIS Field Selection Table (FST) for defining the field entries such as tag, indexing technique, name and format to apply to the field.

The following code snippet from the “Indexer” class shows how Lucene uses the FST of J-ISIS and builds the index using the FST of a specific database :

```
public void setFST(FieldSelectionTable fst) {  
    fst_ = fst;  
    docBuilder_ = new DocBuilder(db_, fst);  
}
```

The “setFST” method takes one parameter i.e. “fst” which is an instance of the FST and this allows passing the entries specified in the FST. In addition, the “docBuilder” object is responsible for building Lucene documents by adding fields into the document based on the entries specified in the FST. The actual creation of a new Lucene index can be seen from the following code snippet of the “Indexer” class:

```
public void createIndex(File idxPath) {
    IndexWriter iwriter;

    /**
     * StandardAnalyzer
     * - is good for most European Languages,
     * - removes stop words
     * - convert to lower case
     */
    try {
        //iwriter = new IndexWriter(idxPath, new StandardAnalyzer(), true);
        Directory fsd = FSDirectory.open(idxPath, NoLockFactory.getNoLockFactory());
        iwriter = new IndexWriter(fsd,
            new StandardAnalyzer(Version.LUCENE_CURRENT),
            true,
            IndexWriter.MaxFieldLength.LIMITED);
        //iwriter = new IndexWriter(idxPath, new StandardAnalyzer(), true);
        iwriter.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

The above "CreateIndex" method takes a file parameter for specifying where the index will be created; an instance of Lucene's "IndexWrite" class have been used for writing a new index into a specified location. The "IndexWrite" takes different parameters such as the directory parameter which tells where the index will be created; the analyzer parameter for specifying which analyzer is used to analyze the document content; a boolean parameter which confirms the creation of the index; and the MaxFieldLength parameter which tells the maximum length of the field. Finally, the application is intended to close the index writer.

### **Lucene Search:**

The following code snippet from the "DbHandle" class in a package "org.unesco.jisis.corelib.server" shows how Lucene is used for searching.

```
public long[] searchLucene(String query) throws DbException {  
  
    try {  
        FieldSelectionTable fst = getFieldSelectionTable();  
        int tCount = fst.getEntriesCount();  
        int[] fieldArr = new int[tCount];  
        for (int i = 0; i < tCount; i++) {  
            fieldArr[i] = fst.getEntryByIndex(i).getTag();  
            //System.out.println(fieldArr[i]);  
        }  
        Search search = Search.createQuery(query, idxPath_.getAbsolutePath(),  
            fieldArr, true);  
        long[] mfns = search.search();  
        System.out.println("searchLucene mfns.length=" + mfns.length);  
        return mfns;  
    } catch (Exception ex) {  
        ex.printStackTrace();  
    }  
    return null;  
}
```

The “searchLucene” method takes one string parameter i.e. the query the end user has entered and it uses the FST for analysing the number of fields that were created in the specific database for looking up the query. For creating the actual query, the “Search” class from “org.unesco.jisis.corelib.server” package is used.

The following method in “Search” class returns the number of records (MFN) that matches the user query and also the instance of the “IndexReader” class is used for searching, reading and return matches to the query.

```
public long[] search() throws Exception {
    // dumpIndex();
    Directory fsd = FSDirectory.open(new File(idxPath_), NoLockFactory.getNoLockFactory());
    //IndexReader reader = IndexReader.open(idxPath_);
    // Read Only open
    IndexReader reader = IndexReader.open(fsd, true);
    Searcher searcher = new IndexSearcher(reader);
    //Hits hits = searcher.search(query_.rewrite(reader));
    AllDocCollector collector = new AllDocCollector();
    searcher.search(query_.rewrite(reader), collector);
    List<ScoreDoc> hits = collector.getHits();
    long docs[] = new long[hits.size()];
    //long docs[] = new long[hits.totalHits];

    // logger.log(Level.INFO, "hits: " + hits.length());
    for (int i = 0; i < hits.size(); i++) {

        Document d = searcher.doc(hits.get(i).doc);
        docs[i] = Long.parseLong(d.get("MFN"));
    }

    reader.close();
    return docs;
}
```

In J-ISIS, saving the content, which is uploaded from a J-ISIS worksheet into a Berkeley DB record, and indexing of the newly added document is done at the same time. This is due to the fact of embedding an instance of the “indexer” class into a “DbHandle” which is responsible for handling database operations, such as adding a record to a database; update a record, retrieving a record, deleting a record and so on. Every time, when we do different operations with our database, we are also building the index of the records in the database by traversing through the records. The following code snippet shows how the index is built using the “indexer” class instance.

```
public boolean buildIndex() throws DbException {  
  
    try {  
        if (!indexer_.fstIsValid()) {  
            try {  
                throw new Exception("There are errors in the FST!");  
            } catch (Exception ex) {  
                ex.printStackTrace();  
            }  
        }  
    }  
  
    /** 2) Create the Documents */  
    indexer_.openWriter();  
    indexer_.tuneWriter();  
    long recordCount = database_.count();  
    if (recordCount <= 0) {  
        throw new Exception("Database is empty!");  
    }  
    DatabaseEntry dbeKey = new DatabaseEntry();  
    DatabaseEntry dbeData = new DatabaseEntry();  
    if (cursor_.getFirst(dbeKey, dbeData, null) == OperationStatus.SUCCESS) {  
        Record record = (Record) Record.createRecord(dbeData.getData());  
        indexer_.writeDocument(record);  
    } else {  
        throw new Exception("Build Index Get First Record Command Failed");  
    }  
    for (long i = 1; i < recordCount; i++) {  
        if (i % 1000 == 0) {  
            System.out.println("Indexing i=" + i);  
        }  
        if (cursor_.getNext(dbeKey, dbeData, null) == OperationStatus.SUCCESS) {  
            Record record = (Record) Record.createRecord(dbeData.getData());  
            indexer_.writeDocument(record);  
            record = null;  
        } else {  
            throw new Exception("Build Index Get Next Record Command Failed i=" + i  
                + " recordCount=" + recordCount);  
        }  
    }  
    indexer_.closeWriter();  
} catch (Exception ex) {  
    ex.printStackTrace();  
}  
return true;  
}
```

From the above code, we understand that for building the index of a database, first we have to make sure that the index writer is opened. In addition, we have to get the record count of the database for building the index. If the record count is zero, an exception with a message “database is empty” will be thrown, otherwise we iterate over all database records using the cursor. The cursor class of Berkeley DB is used for operating on the collections of database records, for iterating over a database, and for saving handles to individual database records (see: [http://docs.oracle.com/cd/E17277\\_02/html/java/com/sleepycat/je/Cursor.html](http://docs.oracle.com/cd/E17277_02/html/java/com/sleepycat/je/Cursor.html)). The

following line from the above code moves the database cursor to the first key/data pair of the database and returns the pair.

```
if (cursor_.getFirst(dbeKey, dbeData, null) == OperationStatus.SUCCESS)
```

If moving the cursor and returning the first key/data pair was successful, the next step will be to retrieve the data from the pair and store it as a record. Finally, the indexer “WriteDocument” method will build the document and add the document into the indexer writer. This process will be iterated using the cursor object.

Lucene provides a way of ranking search results based on the frequency of term occurrences in the document. However, initially, J-ISIS needed to be consistent with the previous generation of ISIS (WINISIS) about how search results are displayed and the ranking feature of Lucene was not activated. Therefore, results were displayed by MFN number sequentially. Later on, we decided to test the ranking capability of Lucene and we have activated the ranking feature by just commenting out a line in a class called “srchTopComponent” in a package “org.unesco.jisis.datasearch”, which was sorting the retrieved search result using array sorting method.<sup>1</sup>

This was done as follows :

---

<sup>1</sup> Ranking in Lucene is still under development and discussion by the Lucene experts, see e.g. [http://mail-archives.apache.org/mod\\_mbox/lucene-java-user/201005.mbox/%3C12y9ab5344d1005050958kd7c63781h8a90229c85a08bb1@mail.gmail.com%3E](http://mail-archives.apache.org/mod_mbox/lucene-java-user/201005.mbox/%3C12y9ab5344d1005050958kd7c63781h8a90229c85a08bb1@mail.gmail.com%3E)

```
private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {
    // JComponent[] fields = {termPanel, keywordBox, optionBox, queryField, plusButton, minusButton};

    String squery = "";
    String userQuery = "";
    StringBuffer buffer = new StringBuffer();
    StringBuffer sb = new StringBuffer(); // Query as entered by user
    if (this.isGuidedSearch_) {
        String join = this.matchAllRadio.isSelected() ? " AND " : " OR ";
        for (JComponent[] fields : this.queryPanels_) {
            // The field tag
            String key = "" + ((SearchableField) ((javax.swing.JComboBox) fields[1]).getSelectedItem()).tag;
            String option = (String) ((javax.swing.JComboBox) fields[2]).getSelectedItem();
            String value = ((javax.swing.text.JTextComponent) fields[3]).getText();

            if (value.equals("")) {
                continue;
            }
            sb.append(key.equalsIgnoreCase("-1")?"Search All Fields"
                : "Search Field "+key);
            sb.append(" "+option+" ");
            sb.append(value);
            sb.append(join);

            squery = this.buildQuery(key, option, value);
            buffer.append(squery);
            buffer.append(join);
        }

        squery = buffer.toString();
        if (squery.endsWith(" AND ") || squery.endsWith(" OR ")) {
            squery = squery.substring(0, squery.length() - 5);
        }
        userQuery = sb.toString();
        if (userQuery.endsWith(" AND ") || userQuery.endsWith(" OR ")) {
            userQuery = userQuery.substring(0, userQuery.length() - 5);
        }
    } else {
        squery = ((javax.swing.text.JTextComponent) this.queryPanels_.get(0)[0]).getText();
        userQuery = squery;
    }
    try {
        //IDatabase db = DatabasePool.getDefaultDatabase();
        if (this.isGuidedSearch_) {
            System.out.println("Guided Search Parsed Query:"+squery);
            results_ = db_.searchLucene(squery);
        } else {

            results_ = db_.searchLucene(squery);
        }
    }
}
```

```
// Sort the array in case
//Arrays.sort(results_);
/* Build the Jlist model from the mfn retrieved */
DefaultListModel model = (DefaultListModel) this.mfnList.getModel();
model.clear();
this.recordLabel.setText(org.openide.util.NbBundle.getMessage(srchTopComponent.class, "results"));
if (this.results_ == null) {
    JOptionPane.showMessageDialog(this, "A syntax error occurred in the query!", "Syntax Error", JOptionPane.WARNING_MESSAGE);
    this.mfnList.setModel(model);
    return;
}
if (results_.length > 0) {
    SearchResult searchResult = new SearchResult(0, db_.getDatabaseName(), userQuery, results_);
    db_.addSearchResult(searchResult);
}

this.maxResults_ = 100;
this.titleBoxActionPerformed(null);

this.recordLabel.setText("" + this.results_.length + " " + org.openide.util.NbBundle.getMessage(srchTopComponent.class, "results").toLowerCase());

if (this.results_.length > 0) {
    this.mfnList.setSelectedIndex(0);
} else {
    JOptionPane.showMessageDialog(this, "No results were found!", "No results found", JOptionPane.INFORMATION_MESSAGE);
}

} catch (Exception ex) {
    ex.printStackTrace();
}
}
```

As can be seen from the above code snippet, the line we commented was:

```
Arrays.sort(results_);
```

As a result of this, we could rank the search result using Lucene.

### 5.1.3.2. How Berkeley DB is Used in J-ISIS

Once a user uploaded file content into J-ISIS worksheet successfully, the next trick to be noted is how J-ISIS will map the file content from the worksheet into a Berkeley DB. After loading a file into J-ISIS worksheet, clicking on the save icon in the worksheet window will save the file content into the file system which was specified as environment database directory during database environment definition. In addition, the index will be built automatically during saving of the record. In this section, we will discuss how mapping of the actual file content and indexing of the content was done. The following code snippet from “org.unesco.jisis.dataentryex.OutlineTopComponent” class shows which action will be invoked when clicking on the save button:

```
private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {  
  
    saveRecord();  
    String msg = NbBundle.getMessage(OutlineTopComponent.class,  
        "MSG_RecordSuccessfullySaved")+ " MFN="+currRec_.getMfn();  
    GuiGlobal.output(msg);  
}
```

When clicking on the save button, the “btnSaveActionPerformed” will invoke “saveRecord” method from the same class and the detail of it can be seen from the following code snippet:

```
private void saveRecord() {  
  
    dataEntryPanel.completeEdit();  
    currRec_ = dataEntryPanel.getRecord();  
    try {  
        currRec_ = db_.updateRecord((Record) currRec_);  
  
    } catch (Exception ex) {  
        Exceptions.printStackTrace(ex);  
    }  
    setRecordChangedFlag(false);  
    setRecord(currRec_);  
    med.dbRecord();  
}
```

The “saveRecord” method is using an instance of “DataEntryPanel” class from “org.unesco.jisis.dataentryex” package. This class is responsible for mapping the content of a J-ISIS worksheet into a tree data structure which has stores data in a key- object pair, in which the tag numbers of fields defined in J-ISIS worksheet will be mapped into “key” of the tree and the content of the record will be mapped into “object” of the tree structure.

```

public IRecord getRecord() {
    Record newRecord = null;
    try {
        // Build a treemap to get fields in increasing tag orders and
        // to build occurrences in a single string
        TreeMap<Integer, Object> fieldTreeMap = new TreeMap<Integer, Object>();
        // Loop on the field nodes
        for (DataEntryNode node = (DataEntryNode) treeModel_.getDataEntryTreeRoot().getFirstChild();
            node != null; node = (DataEntryNode) node.getNextSibling()) {

            Hashtable nodeData = (Hashtable) node.getUserObject();
            int tag = Integer.valueOf((String)nodeData.get("tag"));
            Object obj = nodeData.get("data");
            if (obj instanceof String) {
                String value = (String) obj;

                String fieldValue = (String) fieldTreeMap.get(new Integer(tag));
                if (fieldValue == null) {
                    // 1st occurrence
                    fieldTreeMap.put(new Integer(tag), value);
                } else {
                    // Other occurrences
                    fieldValue += Global.REPETITION_SEPARATOR;
                    fieldValue += value;
                    fieldTreeMap.put(new Integer(tag), fieldValue);
                }
            } else {
                // obj Should be a blob
                // Any occurrences already there?
                ArrayList fieldValues = null;
                fieldValues = (ArrayList) fieldTreeMap.get(new Integer(tag));
                if (fieldValues == null) {
                    // 1st occurrence
                    fieldValues = new ArrayList();
                }
                fieldValues.add(obj);
                fieldTreeMap.put(new Integer(tag), fieldValues);
            }
        }

        if (record_ != null) {
            // We are updating an existing record, transfer in the TreeMap the
            // fields not defined in the worksheet if any
            int nFields = record_.getFieldCount();
            for (int i = 0; i < nFields; i++) {
                IField field = record_.getFieldByIndex(i);
                int tag = field.getTag();
                // Be sure the tag exists in the fdt
                int ifind = fdt_.findField(tag);
                if (ifind < 0) {
                    continue; // Skip
                }
                Object obj = fieldTreeMap.get(new Integer(tag));
                if (obj instanceof String) {
                    String fieldValue = (String) obj;
                    if (fieldValue == null) {
                        // No data in the worksheet, get record field data if any
                        fieldValue = field.getStringFieldValue();
                        fieldTreeMap.put(new Integer(tag), fieldValue);
                    }
                } else {
                    // Should be a blob
                    if (obj == null) {
                        // No data in the worksheet, get record field data if any
                        ArrayList fieldValues = new ArrayList();
                        for (int k = 0; k < field.getOccurrenceCount(); i++) {
                            fieldValues.add(field.getOccurrenceValue(k));
                        }
                        fieldTreeMap.put(new Integer(tag), fieldValues);
                    }
                }
            }
        }
    }
}

```

```
//Build the new record
newRecord = (Record) Record.createRecord();
if (record_ != null) {
    // We are updating the record, transfer the MFN
    newRecord.setMfn(record_.getMfn());
}
Set<Map.Entry<Integer, Object>> set = fieldTreeMap.entrySet();

for (Map.Entry<Integer, Object> entry : set) {
    Integer tag = entry.getKey();
    Object obj = entry.getValue();
    IField field = FieldFactory.makeField(tag);
    if (obj instanceof String) {
        String fieldValue = (String) obj;

        field.setType(Global.FIELD_TYPE_ALPHANUMERIC);
        field.setFieldValue(fieldValue);
        newRecord.addField(field);
    } else {
        field.setType(Global.FIELD_TYPE_BLOB);
        ArrayList values = (ArrayList) obj;
        for (int k = 0; k < values.size(); k++) {
            field.setOccurrence(field.getOccurrenceCount(), (byte[]) values.get(k));
        }
        newRecord.addField(field);
    }
}
} catch (Exception ex) {
    Exceptions.printStackTrace(ex);
}
return newRecord;
}
```

The above “getRecord” method in “DataEntryPanel” class is responsible for returning the record object. This way, the record can be saved by the above “saveMethod”.

The following code snippet shows how a record is added into a Berkeley DB:

```
public Record addRecord(Record record) throws Exception {  
  
    synchronized (this) {  
        Transaction txn = null;  
        long mfn = 0;  
        try {  
            /** start a transaction */  
            txn = database_.getEnvironment().beginTransaction(null, null);  
            mfn = record.getMfn();  
  
            /** DatabaseEntry Encodes database key and data items as a byte array. */  
            DatabaseEntry dbeKey = new DatabaseEntry();  
  
            /** We create a new record -- get next mfn */  
            dbeKey.setData("mfn".getBytes("UTF8"));  
            SequenceConfig seqConfig = new SequenceConfig();  
            seqConfig.setAllowCreate(false);  
            Sequence seq = config_.openSequence(txn, dbeKey, seqConfig);  
            mfn = seq.get(txn, 1);  
            record.setMfn(mfn);  
            seq.close();  
            try {  
                // Record key  
                //dbeKey.setData(longToByte(mfn));  
                // Record data  
                DatabaseEntry dbeData = new DatabaseEntry(record.toBytesEx());  
                database_.put(txn, dbeKey, dbeData);  
                txn.commit();  
            } catch (Exception e) {  
                serverLogger_.error("Exception in addRecord", e);  
            }  
            //logger.finer("DbHandle::addRecord: new Record addeed mfn=" + mfn + record.toString());  
            return record;  
        } catch (DatabaseException e) {  
            txn.abort();  
            serverLogger_.error("DbHandle::addRecord: mfn=" + mfn, e);  
            throw new Exception("Add Record Command Failed: " + mfn + " " + e.getMessage());  
        }  
    }  
}
```

Berkeley DB databases can be transactional. This means, it is possible to specify Berkeley DB to support transactions among multiple operations by specifying this property during creation of the database (Grosso, 2004). As can be seen from the above code, for adding a new record into our database a new transaction has been created using an instance of the environment. This is because, when creating the database instance, it has been defined to support transaction across multiple operations.

In view of the fact that Berkeley DB stores data as a key-value pair in which the keys are indexed and used as identifiers of the value stored in a specific record. These key and value pairs are an instance of a `com.sleepycat.je.DatabaseEntry` Berkeley DB class and this class represents both the key and the data as a `DatabaseEntry` objects. In the above code snippet,

the MFN number was used as key value and the actual content of the record will be the value of the data object. The following line shows how a record is added into Berkeley DB:

```
database_.put(txn, dbKey, dbData);
```

The “database\_” variable refers to the instance of the database. In Berkeley DB, databases with null transaction are automatically committed (Grosso, 2004). Since our database supports transactions, we are responsible to commit the transaction using the following code:

```
txn.commit();
```

#### **5.1.4. Unicode**

Unicode is one of the crucial technical requirements for a digital library, because digital libraries often deal with local cultural heritage, which can be in many different scripts. Due to the fact that J-ISIS and the technologies integrated with J-ISIS (Berkeley DB, Lucene and Tika) are all Java based, they are Unicode compatible since Java is now fully Unicode-compatible. This implies that J-ISIS supports multiple scripts and hence helps to store different resources written in local scripts. The main meaning of this particular J-ISIS feature is that J-ISIS as a digital library could be used for preserving and maintaining artefacts written in local languages and scripts or alphabets. For instance: in Ethiopia there are a lot of historical manuscripts and books written in the local languages (Amharic, Tigrigna, and the almost extinct Ge'ez) and we need to digitize and take care of these resources to not get spoiled only by keeping them in a paper based format. This technique is called 'digital preservation' or preservation by digitization (Haile, 2010). By using J-ISIS we can digitize the resources and store them in a safe way. For this reason, software like J-ISIS will be very important in view of its Unicode support in addition to being under the licence of Free and Open Source Software (FOSS). The following figures are illustration of the fact that J-ISIS can support different scripts.



Tag	Field/Occurrence
1:	<< <a href="#">click here to access the original document</a> >> << <b>Title:</b> 条例须知
2:	<b>Type:</b> application/pdf <b>Author:</b> INS>> <<条例须知条例须知 到达美国入境时访问学者所应注意事项 美国移民海关总署的留学生及访问学者项目 (SEVP) 致力于让您在享用本国之学术、教育和文化资源时为您在美国的停留提供便利。为了在提高安全系数的同时不给合法旅行造成障碍，国家安全局(DHS)在美国入境和出境手续中制定了一些变动。访问学者如果认真计划准备，就能保证将这些手续可能造成的延误降至最低。 为您的到来计划 交换访客不能在 DS 2019 第三条中所注明的开课开始日期30天之前入境。部分访问学者项目的赞助方可能会将提前入境的期限缩短到30天以下。请同您的赞助方确认其具体要求。 随时随身携带您的 护照文件 请不要把以下文件放入寄存行李。如果您的行李遗失或者延误，您将不能在港口提供所需文件。您可能因此而不能进入美国国境。 1. 您的护照，至少在您预计停留时间以后的六个月仍然有效。以及 2. 如果重新进入美国国境，DS 2019 另外，我们强烈建议您也随身携带下列文件： 1. 经济担保证明； 2. 参与访问学者项目的接受函 3. SEVP 前用的支付收据，表格 I-797；以及 4. 赞助单位的名称及联络方式，包括紧急状况下24小时有效的联络电话。 关于在美国入境和旅行手续的更多细节，请访问： <a href="http://educationusa.state.gov/predeparture/travel/customs.htm">http://educationusa.state.gov/predeparture/travel/customs.htm</a> 填写您的入境表格  如果航空到达：飞机乘务员将分发海关申报表(CF-6059)和出入境记录表(I-94)。这些表格必须在着陆前填写完毕。 如果陆路或航路到达：港口的CBP官员将提供所需海关申报表(CF-6059)和出入境记录表(I-94)，在您到达时填写。 在您到达入境口时 前往为到达乘客准备的终端区域。准备出示下列文件：您的护照；装有您DS-2019表格的密封信件；出入境记录表(I-94)；以及 海关申报表(CF-6059)。I-94表格应当显示您在美国的居住地点，而不是赞助项目的地址。 所有进入美国的访客都必须陈述其入境理由。您也将被要求提供您最终目的地的相

Figure 13: Chinese Document in Web-JISIS after Having Done a Search with Highlighted Search-keys

1:	<p>• ሁሉ ቀን መመዘኛ ከብሪቲሽን መመዘኛ፣ ከብሪቲሽ ሲጅድር ሃኪምን ግለታዎች</p> <p>• ጭው የበላባት ምክብ ለሰጠላት</p> <p>• የሚያገለግሉ ከሆኑ ሲጋራ ግብዓት</p> <p>• የልሳሳ ምጣኑ ለሰጠላት</p> <p>• በዕለቱ የፍትህ ክትትል መውሰድ</p> <p>• የሃኪምን ቅጠር ሁሉ ግክብር</p> <p>ለሃኪም ማደግ ያለብኝ ምን ሲሆን ነው?</p> <p>• የትንቢት ለጥረት ከተገባለ</p> <p>• ደረጃ ላይ ህመም ሲለግድ</p> <p>• ከብሪቲሽ ወደላይ ከወጣ</p> <p>• ፎቀት ከተገባለሁ</p> <p><a href="http://ethnomed.org">http://ethnomed.org</a> What is Heart Failure? - Amharic September 2011 Page 1 of 1</p>
2:	<p>What is Heart Failure?</p> <p>Heart failure means the heart muscle has become weak and does not pump enough blood for the rest of your body. It does not mean that the heart stops.</p>

Figure 14: Illustration of J-ISIS Capable of Handling Mixing Scripts: Case of Amharic and English

## **5.2. Implementation of the Digital Library Feature**

In addition to the previously mentioned and briefly discussed technical requirements for J-ISIS, which are already available in the J-ISIS software but needed some more testing, there are features that need to be addressed in the remaining part of this thesis. One of these, i.e. an interface for incorporating documents with their full-text into the system, will be discussed in full detail since it represents our main practical contribution to J-ISIS, involving coding, testing and improving. This took a major part of the time investment as a lot had to be learned about both the existing code of J-ISIS and about how to code such new feature. Especially in view of the need to keep this implementation in line with the ISIS-philosophy of flexibility. Finally, some few remaining techniques for digital libraries will be briefly mentioned for completeness' sake and referral to future work.

This section is organized in the following way:

- In section 5.2.1, the newly added digital library feature into J-ISIS will be presented.
- In section 5.2.2, technical discussion of the newly implemented feature will be presented.
- In section 5.2.3, some results of testing J-ISIS performance in comparison with Greenstone digital library software will be presented.

### **5.2.1. Presentation of the Newly Added Digital Library Feature**

The current version of J-ISIS is not fully ready and applicable for practical uses as there are still features which have to be implemented and addressed. One of the crucial features which I wanted to implement as a subject of this paper is a digital library feature. This will enhance the capability of J-ISIS and acts as our main initiative of practical implementation with J-ISIS.

For implementing digital library technology into J-ISIS, the most important contribution needed was to design and implement an easy interface for the system managers or librarians to actually load the documents into the collection or database.

To design the new feature, we used the Greenstone and ABCD approach of such feature into these systems.

The Greenstone digital library System is a software in FOSS-setup designed from scratch by the Waikato University in New-Zealand for the purpose of digital libraries. It is used very widely nowadays in many projects and is supported by UNESCO. The technology is based on Java for the client (the 'Librarian's Interface'), Perl for the web-scripts, XML for the storage of the text-contents of the documents and either its own full-text indexer (MGPP) or Lucene (Witten et al., 2001).

From Greenstone we adopted the idea of:

- allowing pre-view of the text-contents of the documents along with the URL for opening the original document in its full format (e.g. PDF);
- the use of plug-ins to deal with many different document formats and their text-extraction.

The ABCD implementation of digital library (Berhe, 2011) is built on the same ideas: a fast and small-footprint PDF-to-Text converter (PDF2HTML.exe) for PDF's and a command-line Tika Java-executable for doing the other common document types. Whereas Tika can do all types of documents, most of all PDF, the output of PDF2HTML is more attractive once presented in a web-page and its execution is much faster due to being a small executable to be loaded in memory. This new ABCD digital library feature automatically uploads the document to the Apache domain on the server, and creates the URL automatically as a hyperlink in the record, along with full-text indexing of the extracted text-contents.

Another important idea of the design is that the feature should fully integrate into the ISIS-approach: the librarians, not the software, define the implementation of the feature by defining which fields to use for which purpose. In the case of digital library, we considered two fields as being crucial: the URL and the document text. The librarian should be allowed to define which fields to use to store these particular values. By doing so, the digital library feature would also be perfectly compatible and easy to merge into existing or newly created metadata structures.

Incremental indexing, which means that immediately after loading a document into the database the new text will be indexed and made searchable is another feature on our wish-list. In Greenstone with its standard indexer e.g. this is not possible and in ABCD we could only

implement this with the more restricted version of the standard ISIS-software which has severe document-size restrictions.

Finally, some considerations were about a reasonable speed: (uploading/loading a bigger document should not take more than one minute, and so does the indexing of a document).

#### **5.2.1.1. Creation of New Field Types**

J-ISIS, as a database software stores data as records. A record can have one or more fields of different types such as alphanumeric, alphabetic, numeric, pattern, string, Boolean, char, byte, short, int, float, long, double, blob, date and time. For each field type a unique number is assigned to identify each field type. In addition to the existing field types of J-ISIS, to store documents and preserve a link to the original content of the document, it was necessary to create two new field types i.e. URL and DOC. The reason why we need to preserve a link to the original content of a document is, in digital library applications, “what you get is not what you stored” (Arms, 1995; Vijayakumar and Jeevan, 2001). This means, in J-ISIS when we extract text content of a document it will lose its original look and feel. For example, images will be omitted by Tika. Therefore, it is crucial to allow access to the original document by preserving a link to it.

For creating the fields related to digital library, in J-ISIS-core module within the “Global.java” class the field types have been introduced as follows:

```
public class Global {  
    private static String clientWorkPath_ = null;  
    private static String clientTempPath_ = null;  
  
    public final static String[] fieldTypes = new String[]{  
        "alphanumeric",  
        "alphabetic",  
        "numeric",  
        "pattern",  
        "string",  
        "boolean",  
        "char",  
        "byte",  
        "short",  
        "int",  
        "float",  
        "long",  
        "double",  
        "blob",  
        "date",  
        "time",  
        "url",  
        "doc"  
    };  
};
```

Moreover, the following code snippet shows that how a number is assigned to these field types:

```
public final static int FIELD_TYPE_URL = 16;  
public final static int FIELD_TYPE_DOC = 17;
```

This means, the number 16 is assigned to field type “URL” and the number 17 is assigned to field type “DOC”. So that, whenever the fields are selected, the numbers 16 and 17 will be assigned respectively to “URL” and “DOC”. Moreover, the following code snippet shows how these newly added field types are incorporated with FDT of J-ISIS.

```
private void btnAddEditActionPerformed(java.awt.event.ActionEvent evt) {
    int tag = Integer.parseInt(spinnerTag.getValue().toString());
    String name = txtName.getText();
    String sType = (String) cmbType.getSelectedItemAt();
    int type = 0;
    if (sType.equals("ALPHANUMERIC")) {
        type = Global.FIELD_TYPE_ALPHANUMERIC;
    } else if (sType.equals("ALPHABETIC")) {
        type = Global.FIELD_TYPE_ALPHABETIC;
    } else if (sType.equals("NUMERIC")) {
        type = Global.FIELD_TYPE_NUMERIC;
    } else if (sType.equals("PATTERN")) {
        type = Global.FIELD_TYPE_PATTERN;
    } else if (sType.equals("DATE")) {
        type = Global.FIELD_TYPE_DATE;
    } else if (sType.equals("TIME")) {
        type = Global.FIELD_TYPE_TIME;
    } else if (sType.equals("BLOB")) {
        type = Global.FIELD_TYPE_BLOB;
    } else if (sType.equals("URL")) {
        type = Global.FIELD_TYPE_URL;
    } else if (sType.equals("DOC")) {
        type = Global.FIELD_TYPE_DOC;
    }
}
```

When a user creates a database, the Field Definition Table (FDT) of J-ISIS has to be used for specifying the entry of the fields in a record. This includes the tag number of the field, the name of the field, the type of the field, repeatability (e.g. in case of e-mail address a user could have more than one e-mail and for this purpose the field should be repeatable), e.t.c. This is illustrated as follows:

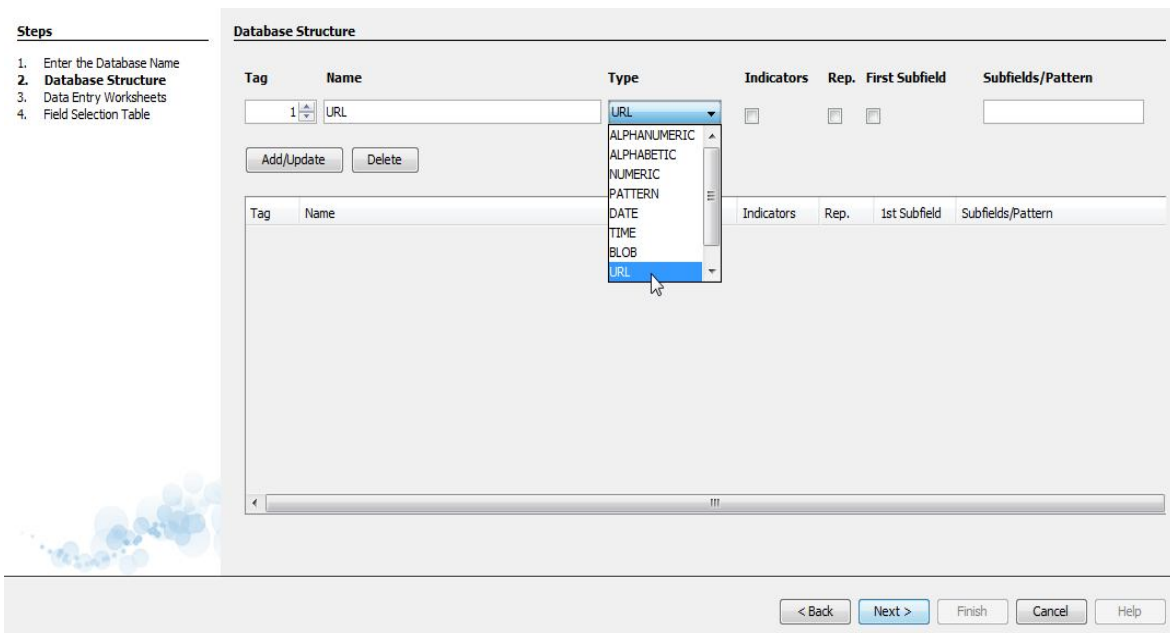
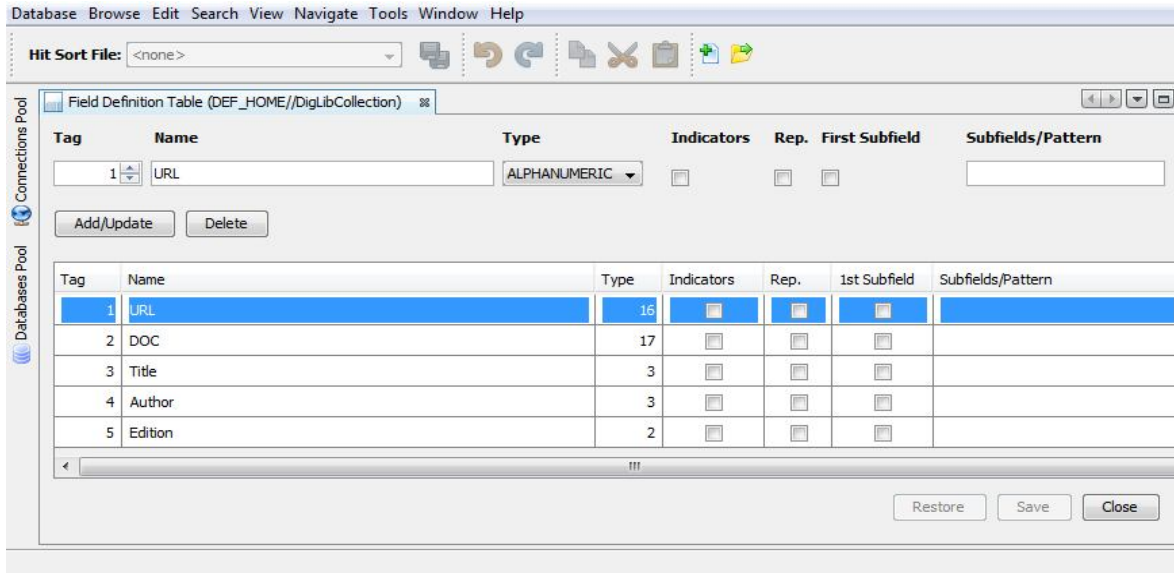


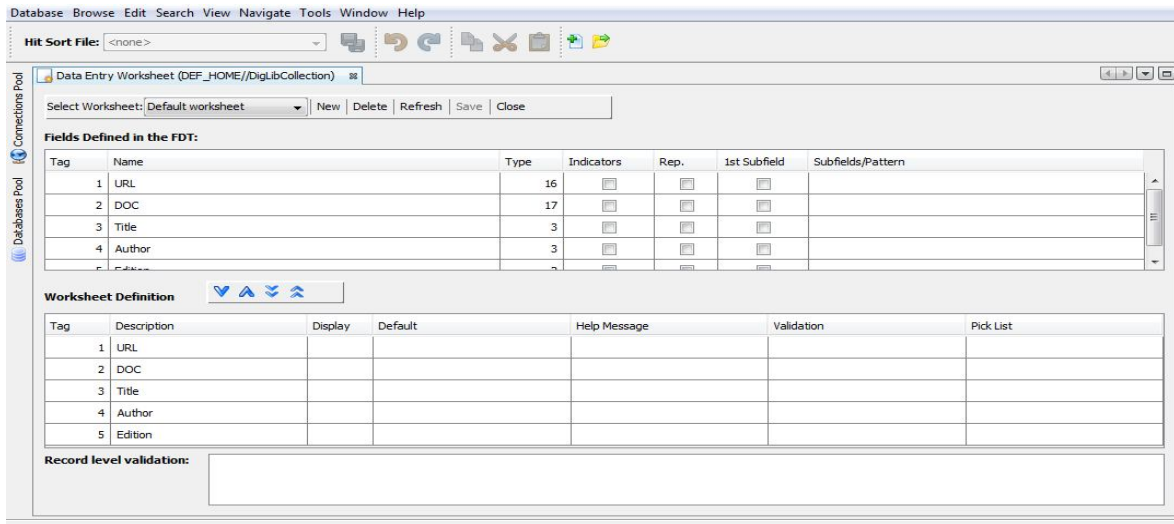
Figure 15: Illustration of the New Added Field Types

After selecting the field entries, clicking on the add/update button will add the fields into the FDT. The following figure shows the FDT of J-ISIS with the two new field types (URL, DOC) and combined with other field types:



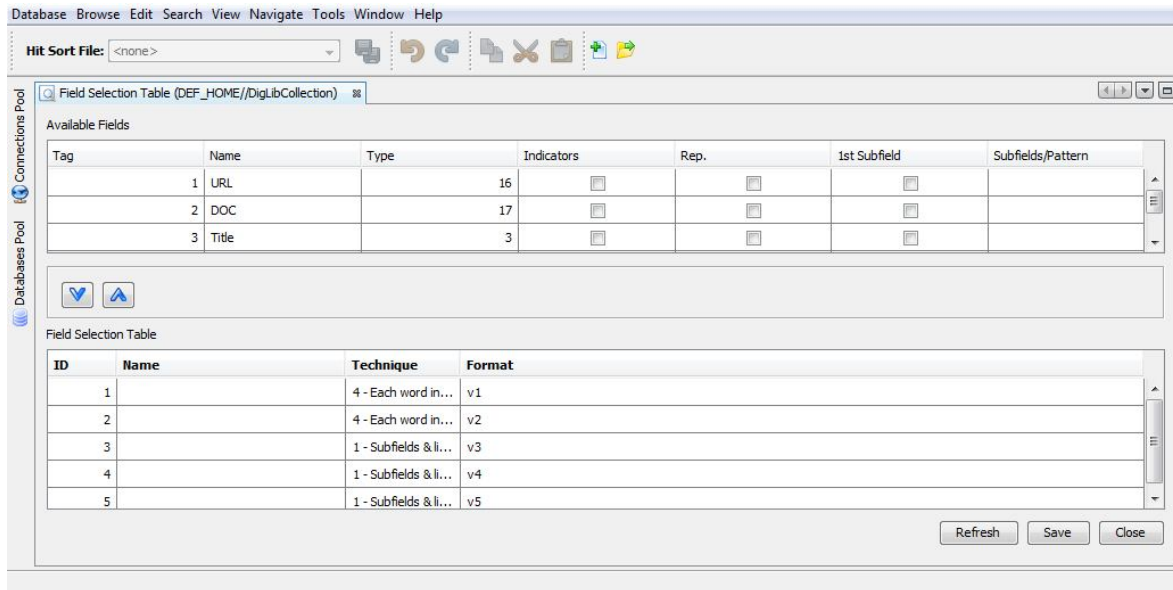
**Figure 16: J-ISIS Field Definition Table Contains Both Normal and Digital Library Field Types**

After adding fields into FDT, the next step will be to add the fields defined in FDT into the Data Entry Worksheet as follows:



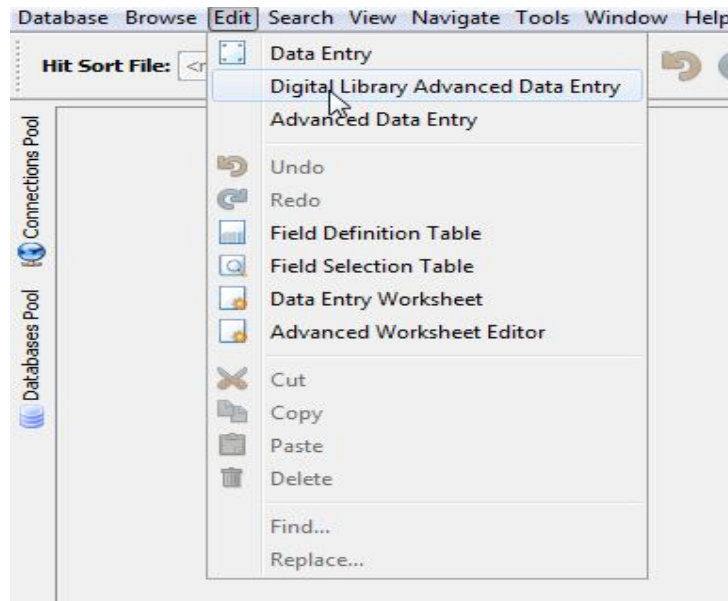
**Figure 17: J-ISIS Data Entry Worksheet Definition**

By adding the fields into the Data Entry Worksheet, we are making sure that the fields will be available during data entry. If it is not added at this stage, it will not be possible to enter data into that particular field. The next step is selecting fields for indexing and this is done in FST of J-ISIS as follows:



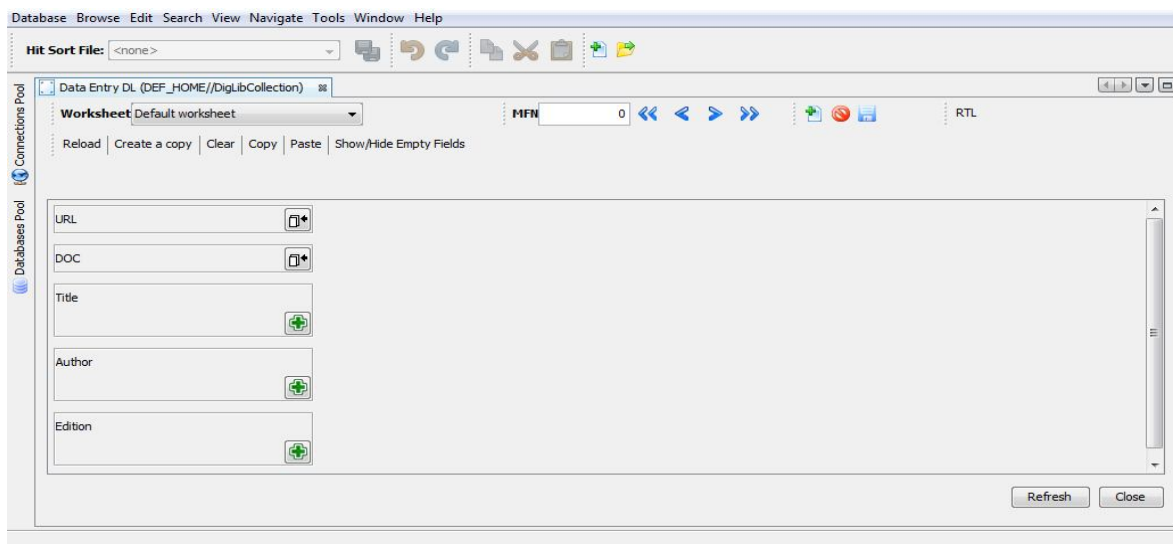
**Figure 18: Selection of Fields in J-ISIS Field Selection Table**

Some steps have been done in the above section to show how a J-ISIS database could be created with the newly added field types. After introduction of the new field types, i.e. URL or DOC, the application will parse the Field Definition Table of J-ISIS and check if any of these new types are defined. Therefore, if one of these field types were used, a special icon will be added into the interface (the J-ISIS worksheet) for uploading the link path of a document or the content of the document depending on the type of the field used. This ensures that a user will not be forced to use a pre-defined field number in a record either to create a hyperlink or to load the actual content of the document. So, a dynamic approach has been used for creating the URL and to upload a document. This point promotes the "free schema" principle. To test the digital library feature, specifically the loading of different document formats into a record of J-ISIS, a new interface element with a name "Digital Library Advanced Entry" was introduced in the existing "Edit" menu of J-ISIS as follows :



**Figure 19: Integration of the New Digital Library Feature and J-ISIS**

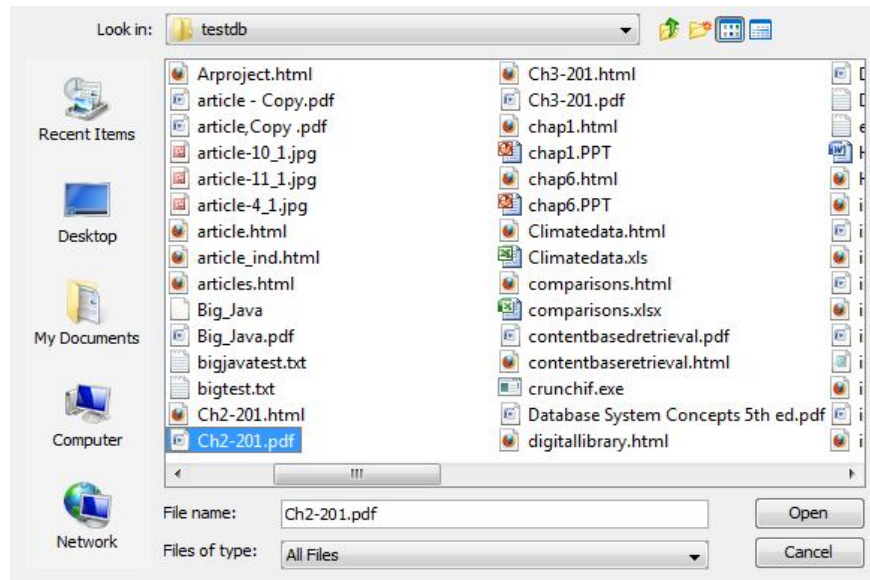
Once the connection to the server is started and a database is opened, clicking on the "Digital Library Advanced Data Entry" menu option will display the following J-ISIS worksheet interface. This interface allows the user to enter data to the database, update data, delete and also navigate through the records of the database.



**Figure 20: Illustration of the Resulting Interface for Traditional and Digital Library Creation**

In addition, the above figure shows that the feature can now easily be integrated into the existing data-entry interface of J-ISIS and no separate approach for classic field structures is necessary.

When a user clicks on the upload icon, a file browser which allows selecting a file for uploading will be displayed as follows:

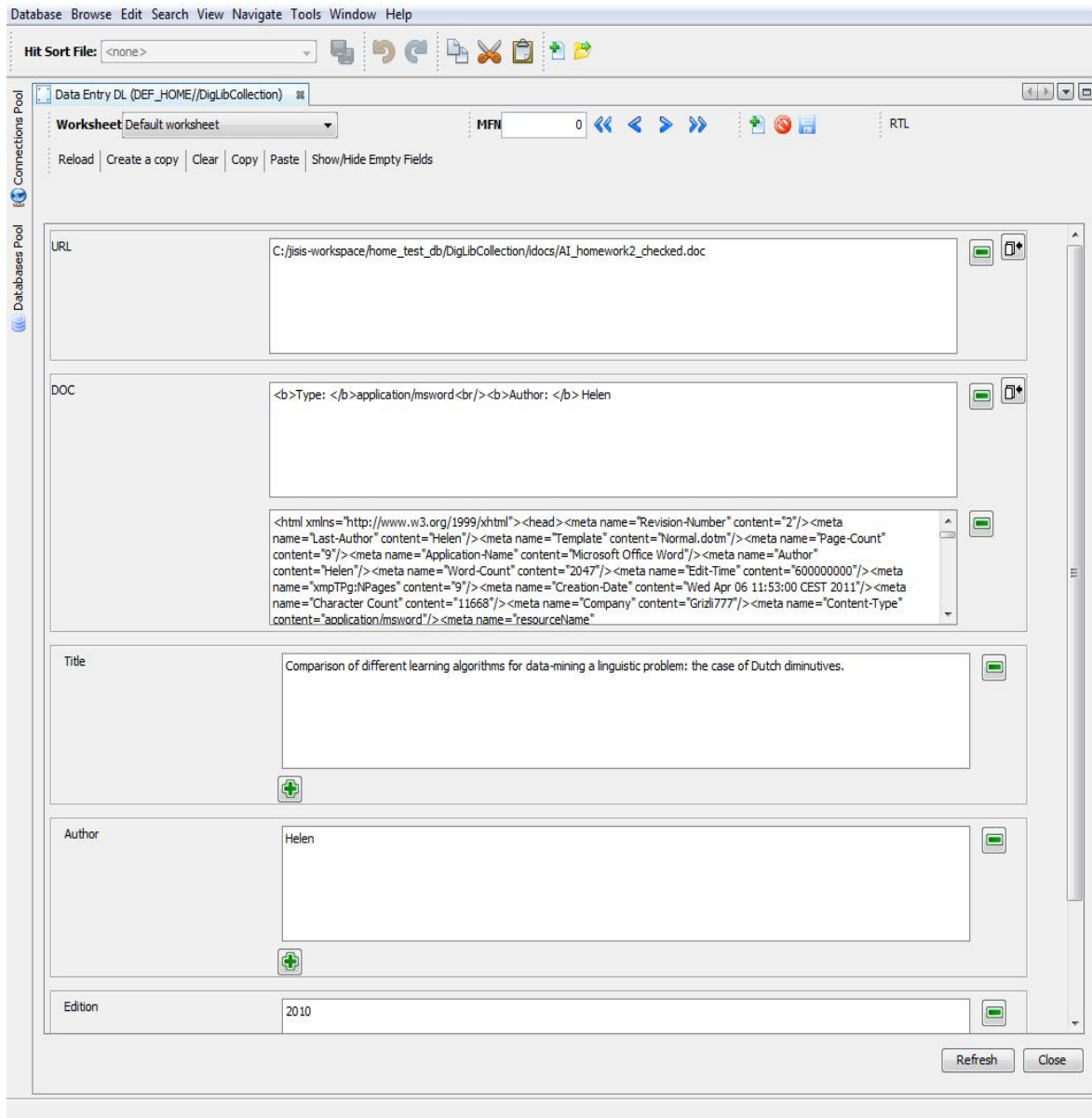


**Figure 21: File Browser for Uploading a Document into J-ISIS Worksheet**

The above file browse interface was designed using the Java file chooser API and it was coded as follows:

```
private File getUriFile() {  
  
    File f = null;  
    JFileChooser fileChooser = new javax.swing.JFileChooser(path);  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("doc", " (MS Word Document)"));  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("pdf", " (Adobe Portable Document Format)"));  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("docx", "MS docxfiles"));  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("xls", " (MS Excel Document)"));  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("ppt", " (MS PowerPoint Document)"));  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("rtf", " (Rich Text Format)"));  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("html", " (HTML Format)"));  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("xhtml", " (XHTML Format)"));  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("odf", " (OpenDocument)"));  
    fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("txt", " (Plain Text)"));  
  
    fileChooser.setSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);  
    fileChooser.setDialogTitle("Select the file to load");  
    fileChooser.setAcceptAllFileFilterUsed(true);  
    Dimension dialogSize = fileChooser.getPreferredSize();  
    Dimension frameSize = getSize();  
    Point loc = getLocation();  
  
    fileChooser.setLocation((frameSize.width - dialogSize.width) / 2 + loc.x, (frameSize.height - dialogSize.height) / 2 + loc.y);  
    if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {  
        f = fileChooser.getSelectedFile();  
        path = f.getPath();  
    }  
    return f;  
}
```

Once a user selects a file of any format for uploading into a J-ISIS record, Tika is used for extracting the text content from various document formats (all major formats are supported) and to load the content into J-ISIS worksheet. The actual uploading of a document into J-ISIS worksheet is illustrated as follows:



**Figure 22: Document Content Loaded into J-ISIS Worksheet**

After saving the content loaded into a database, it will also be indexed based on the provided indexing techniques. This way, one can search the content of the document as follows:

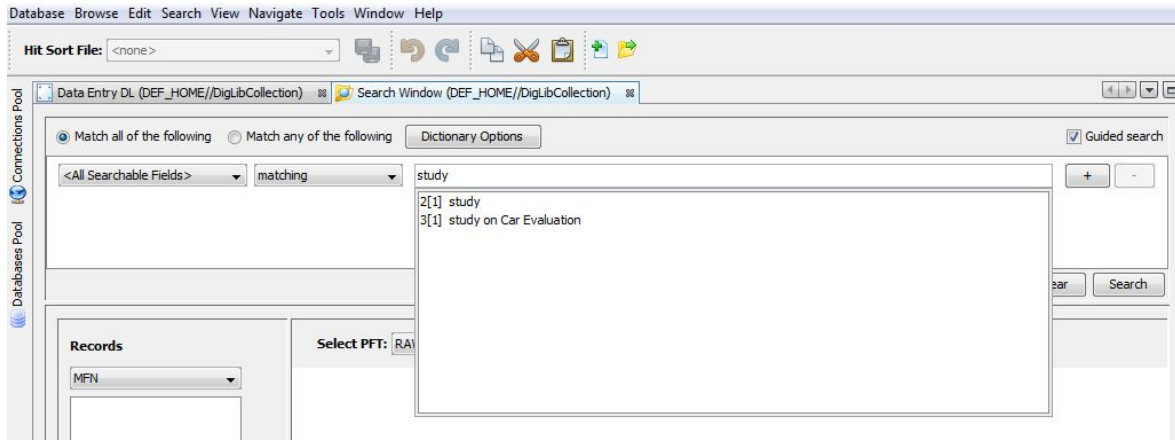


Figure 23: Entering a Search Key into J-ISIS Search Interface

The following figure shows the result of searching the word “study”:

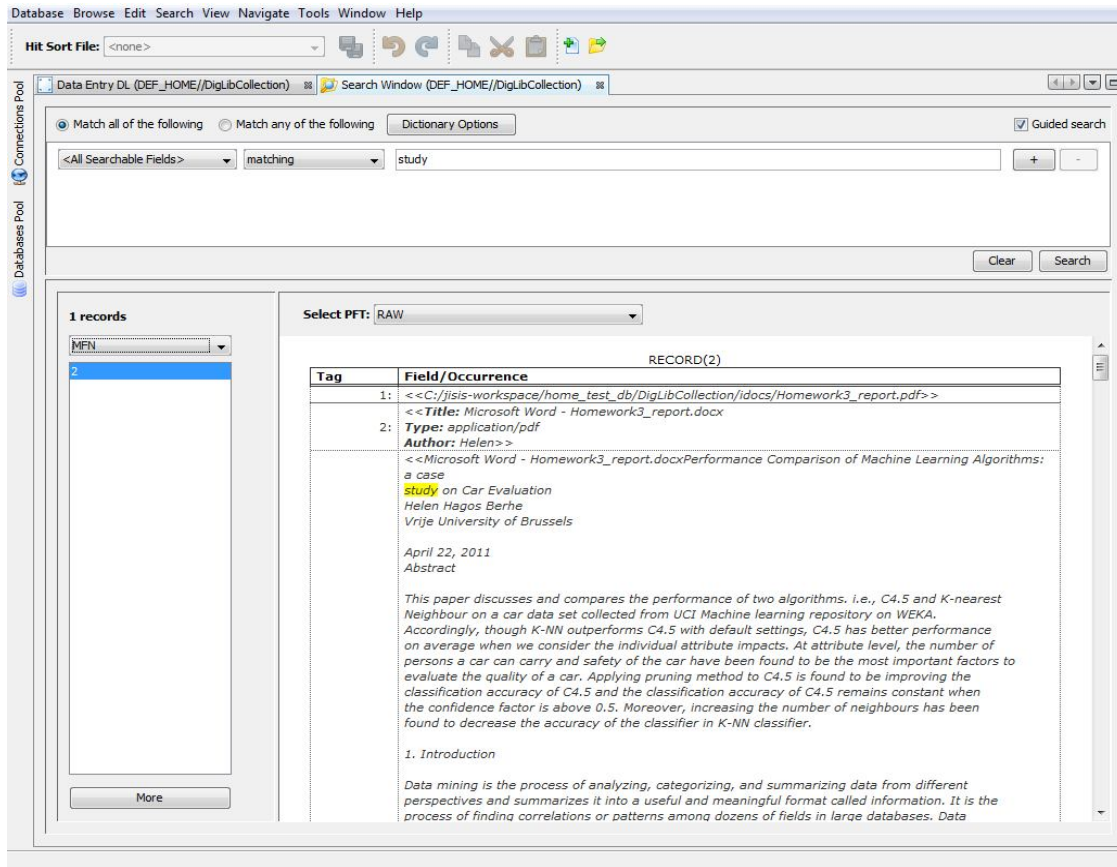
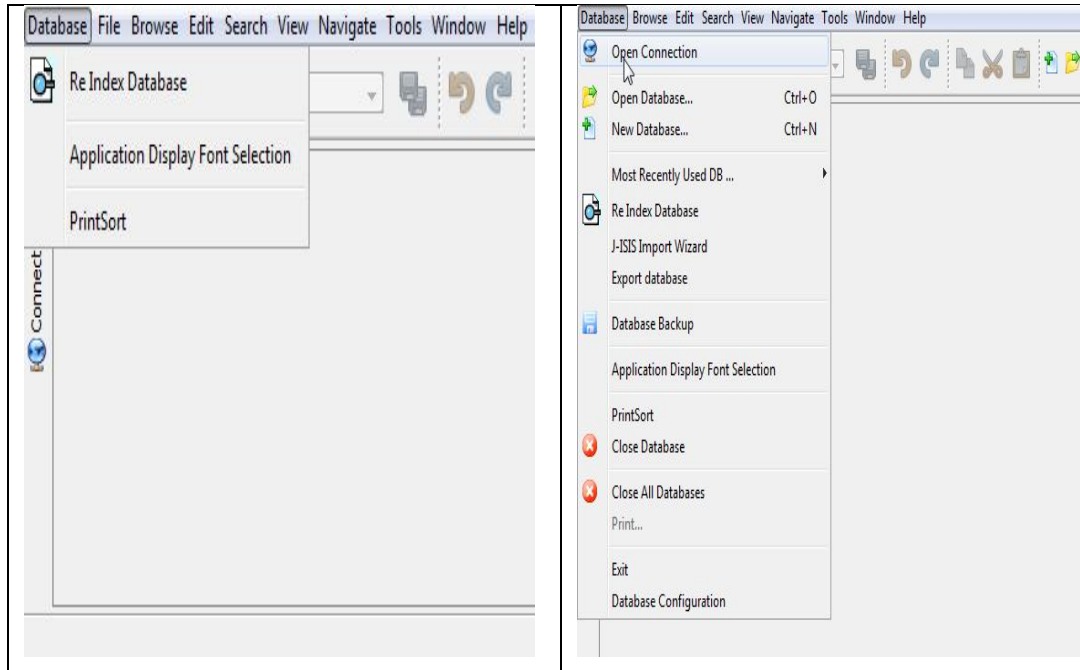


Figure 24: Illustration of the Search Result from J-ISIS Interface

### **5.2.2. Technical Discussion of the Newly Implemented Digital Library Feature**

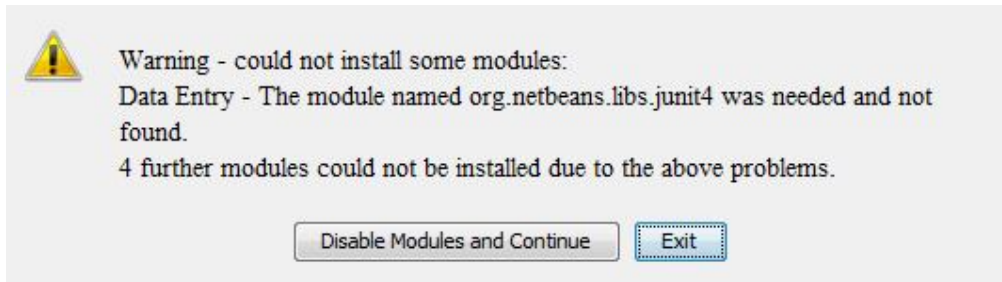
For implementing the digital library feature into J-ISIS, the first step was to get familiar with the J-ISIS software by just running the .exe file of the software and exploring it. Different experiments were done to see which features were already available and which were not there. But later it was needed to get the source code of the software in order to really start the actual work. In spite of having been published in a website for Java-based projects (<http://kenai.com/>), the published version was an older version and it was recommended by the main developer of J-ISIS (Mr. Dauphin, formerly of UNESCO) to wait for getting a more recent version of the software from him. As a result, there was some delay, but finally thanks to Mr. Dauphin for delivering his updated source code and the actual work could start from this point onwards.

However, a first hiccup was a problem we experienced when building the project and trying to run it, even without any changes: due to an incompatibility of NetBeans with Windows7 (something which we found out after long searches and tests), which was reported in the Internet forums but not yet resolved, the menu-structure appeared wrongly and incompletely: the existing main functions of opening a connection to the server and a database were not available and this prevented all further work. Hence experimenting with J-ISIS was not easy as expected to be. Different methods have been tried to solve the problem but remained without success. The following figure illustrates the problem we experienced with menu building:



**Figure 25: J-ISIS with Menu Building Problem**      **Figure 26: J-ISIS with Correct Menu Building**

In addition, building J-ISIS from NetBeans had a problem and displays the following error message:



**Figure 27: J-UNIT Related Error Reported in Netbeans**

We circumvented this problem for the first weeks, until a real solution was found, by running J-ISIS into a Virtual Machine Windows XP. Unfortunately, this was not a very practical and slowly running solution on our PC as it was too slow to do different tasks with J-ISIS. Thus, further investigation into the problem was necessary and finally the problem was found to be related to the different locations for ‘plug-ins’ to be stored, which has changed in Windows7, using a different folder setup for the user-folders (as compared to the ‘Documents and Settings’ setup of earlier Windows versions) and program files. Moreover, there was an inconsistency on how on the one hand NetBeans and on the other hand JDK dealt with this in

Windows7. Modules such as DataEntry, Database Pool and Database wizard, which are important for building the J-ISIS menu system, were deactivated because of the dependencies with the missing JUNIT4 as given in the above error message. The main developer of J-ISIS is working on MacOS and did not experience these problems, proving it was related to the OS used. Cross-platform development has its price.

The problem about “JUNIT not found” was solved by removing the JUNIT dependency in the Data Entry module which was not actually used for the purpose. Once the JUNIT library was removed from Data Entry module dependency, the error has gone and the menu system of J-ISIS could build from the main Operating System.

The main point behind this is, one should give attention to the sometimes tricky combination of the OS, the JDK and IDE being used for building J-ISIS.

Another smaller problem causing some delay in our work was the fact that J-ISIS source codes were available with the J-ISIS-corelib classes only as a .jar, which means the code of the basic classes were not available. We could only really start understanding the whole mechanisms of the database software after getting these codes directly from the main developer and studying also these underlying methods also in the “core-lib” (basic libraries of J-ISIS), e.g. of database storage in Berkeley DB and indexing with Lucene.

After the availability of the source code and building the project was finally secured and proved to run correctly, the analysis of the software aimed to understand the coding of the software and to identify the relevant parts for implementing the new digital library technology. The application was found to be quite huge and for that reason a lot of time was invested on understanding and identifying of the relevant portions which are necessary for implementing the new digital library feature.

J-ISIS was not designed with a digital library feature in mind and no one has tested whether the technologies incorporated with it could really do what they supposed to do. The idea of digital library support with J-ISIS was not addressed at all at the time my thesis took off. This means that at this stage J-ISIS could only be used for a traditional library and other semi-structured database applications. For instance, storing of the metadata of the resources, which gives information about where the resources can be found, who is the author of the resources,

when was the publication year and etc could be done fully but not handling of the documents within the database as digital resources themselves.

### **Tika and J-ISIS Integration**

Tika is a Java-based toolkit which is used to extract metadata and structured text content from various documents. Tika provides a mechanism for hiding the complexity of different document formats and parsing libraries. As a result, Tika provides a simple and powerful API for client applications in order to deal with various document formats. Tika's API is easy to use and extend by adding parsers which are not included with in Tika. As a result, Tika will be able to extract structured text and metadata from the newly added parser (Tikhonov and Mattmann, 2010).

According to Tikhonov and Mattmann (2010), the "org.apache.tika.parser.Parser" interface is the crucial part of Apache Tika which encapsulates various parser libraries and handles the extraction of text and metadata from various file formats. This is performed by using a single method called "parse" and the following code snippet illustrate the arguments used by the method:

```
public void parse(InputStream in, ContentHandler ch, Metadata mtdt, ParseContext pc) throws IOException, SAXException, TikaException;
```

From the above, the first argument is the input stream which reads the document for parsing and due to some reason if the input document cannot be read, the IOException will be thrown. Tika returns the parsed content of the document as XHTML SAX events. In this case, the structured content of the parsed document will be expressed as XHTML and this is used to give structural information of the content and SAX events will be used for processing of the streamed content.

As a result, the processed SAX events will be passed to the content handler which is used as a second argument of the "parse" method. The SaxException will be thrown in case the SAX events could not be passed successfully to the defined ContentHandler. The third argument of the "parse" method is the document metadata, which extracts the metadata from the parser. The last argument is to be used in case of some additional information needed to be informed to the parser. If the document cannot be parsed due to some reason the TikaException will be thrown.

Tika supports the following document formats:

- HyperText Markup Language,
- XML and derived formats,
- Microsoft Office document formats,
- OpenDocument Format,
- Portable Document Format,
- Electronic Publication Format,
- Rich Text Format,
- Compression and packaging formats,
- Text formats,
- Audio formats,
- Image& Video formats; and
- Java class files,
- Archives; and
- the mbox format

(see: <http://tika.apache.org/1.0/formats.html> ).

Tika played an important role for implementing the new digital library feature by providing a ready-made mechanism to extract the text content and metadata of various document formats. The following code snippet shows how Tika was integrated into our digital library data entry interface.

```
{
    Metadata metadata = new Metadata();
    metadata.set(Metadata.RESOURCE_NAME_KEY,
        f.getCanonicalPath());

    InputStream is = new FileInputStream(f);
    Parser parser = new AutoDetectParser();
    //ContentHandler handler= new BodyContentHandler();
    SAXTransformerFactory factory = (SAXTransformerFactory) SAXTransformerFactory.newInstance();
    TransformerHandler transformerHandler = null;
    try {
        transformerHandler = factory.newTransformerHandler();
    } catch (TransformerConfigurationException ex) {
        Exceptions.printStackTrace(ex);
    }
    transformerHandler.getTransformer().setOutputProperty(OutputKeys.METHOD, "html");
    transformerHandler.getTransformer().setOutputProperty(OutputKeys.INDENT, "no");
    transformerHandler.getTransformer().setOutputProperty(OutputKeys.ENCODING, "utf-8");
    StringWriter sw = new StringWriter();
    transformerHandler.setResult(new StreamResult(sw));
    try {
        try {
            is = new ProgressMonitorInputStream(
                this, "Parsing stream", is);
            parser.parse(is, transformerHandler, metadata, new ParseContext());
            String xml = sw.toString();
            String title = metadata.get(Metadata.TITLE);
            String contenttype = metadata.get(Metadata.CONTENT_TYPE);
            String author = metadata.get(Metadata.AUTHOR);
            System.out.println("tika xml output" + xml);
            fld_.setOccurrence(fld_.getOccurrenceCount(), "<b>Title: </b>" + title + "<br/>"
                + "<b>Type: </b>" + contenttype + "<br/>" + "<b>Author: </b>" + author);
            redraw();
            fld_.setOccurrence(fld_.getOccurrenceCount(), xml);
            redraw();
            this.applyComponentOrientation(orientation_);
        } catch (DbException ex) {
            Exceptions.printStackTrace(ex);
        } catch (SAXException ex) {
            Exceptions.printStackTrace(ex);
        } catch (TikaException ex) {
            Exceptions.printStackTrace(ex);
        }
    } finally {
        is.close();
    }
}
```

When Tika parses various documents to text, it generates XHTML SAX events. The way how it will be viewed in the browser should be defined by the application. This is implemented as follows:

```
SAXTransformerFactory factory = (SAXTransformerFactory) SAXTransformerFactory.newInstance();
TransformerHandler transformerHandler = null;
try {
    transformerHandler = factory.newTransformerHandler();
} catch (TransformerConfigurationException ex) {
    Exceptions.printStackTrace(ex);
}
transformerHandler.getTransformer().setOutputProperty(OutputKeys.METHOD, "html");
transformerHandler.getTransformer().setOutputProperty(OutputKeys.INDENT, "no");
transformerHandler.getTransformer().setOutputProperty(OutputKeys.ENCODING, "utf-8");
StringWriter sw = new StringWriter();
transformerHandler.setResult(new StreamResult(sw));
```

If a plain text output is needed for content extracted by Tika, the following code could be used instead of the above code, in this case there will not be any structure applied and we will not need the SAX parser for such text.

```
ContentHandler handler= new BodyContentHandler();
```

The actual parsing process will be done as follows:

```
parser.parse(is, transformerHandler, metadata, new ParseContext());
```

As mentioned above, the arguments given to the "parse" method works as follows: "is" passes the file needed to be parsed to the parser; the transformerHandler handles the content and it will give the XHTML result, the metadata gives the available metadata of the document parsed and the ParseContext will be used in case additional information is needed to be passed to the parser.

The metadata retrieved from metadata object and the content of the documents is loaded into a J-ISIS worksheet as follows:

```
fld_.setOccurrence(fld_.getOccurrenceCount(), "<b>Title: </b>" + title + "<br/>"
    + "<b>Type: </b>" + contenttype + "<br/>" + "<b>Author: </b> " + author);
redraw();
fld_.setOccurrence(fld_.getOccurrenceCount(), xml);
redraw();
```

As a result, the text extracted by Tika has been used as input for Lucene to index these documents and make them ready to be searched and accessible for the end user.

Moreover, in order to allow a user to have access to the original document and for technical reason, we keep both the extracted text and the original document into a J-ISIS server sub-folder i.e. "idoc". The reason to keep the extract text in a server folder was to be able to

upload the text into J-ISIS worksheet from hard disk rather than directly from memory, as uploading from memory resulted into a "java.lang.OutOfMemoryError: Java heap space" error, when uploading really big documents (e.g. e-books). To solve this problem, the first trial was to allow growing the maximum heap space size up to 1GB by setting it in NetBeans.conf file in the directory where we installed NetBeans using the "-J-Xmx1024m" command. This didn't really solve the problem because the actual problem was the total physical availability of memory for the application running.

Since the above solution wasn't solving the problem, another solution was proposed which was, instead of using the extracted text directly from memory, we first stored the text content into a file on the hard disk and then we loaded it back into a J-ISIS worksheet. To some extent, this solution solved the problem. By implementing this solution, we tried to load files ranges from small size up to maximum of several megabytes. In this case a problem appeared when we tried to load files with above 200MB and with text content of about 3MB: it became very slow. This was the issue that explained why storing the text document into the hard disk and then uploading it back into a record was not a good solution. We needed to have a 'reader' function which reads the text content from the hard disk line by line, and this was at the cost of time, making J-ISIS slow and after some time causing it essentially to stop working. For loading a text content of e-book (with text-only size of above 2MB), it took about 38 minutes. This was practically unacceptable from a technical as well as end-users point of view. For instance, imagine having hundreds of e-books in the digital library collection: if it took 38 minutes for one book, how long will it take for having such a bigger collection loaded? This was a big challenge, as we wanted to make J-ISIS a fast, powerful digital library software and digital libraries are not only about documents of a few pages, but they should also handle big documents (this possibility already offered by Berkeley DB which could store any document-size).

We continued our investigation into this problem to find out a real solution which could avoid the "java.lang.OutOfMemoryError: Java heap space" problem fully regardless of document size. Finally, we have learned that the above problem was not only because of the physical memory availability but also one method was found to cause the problem: one existing Java class used in the J-ISIS core libraries had a bug with the setText (text) method, which actually stores the text value into a field of a record, and this allocates large memory space. This was replaced by the following code and the problem was solved:

```
try {  
    this.read(new StringReader(text), "FIELD OCC");  
} catch (IOException ex) {  
    Exceptions.printStackTrace(ex);  
}
```

Another problem we encountered was about Tika's size limitation to 100KB which was giving "org.apache.tika.sax.WriteOutContentHandler\$WriteLimitReachedException: Your document contained more than 100000 characters, and so your requested limit has been reached" error message. Without giving this error message, all documents after extraction were truncated to 100KB whereas the documents were actually of different size, some of them were medium sized and others were quite big. By disabling Tika's content handler default size limitation (setting to size -1) as follows:

```
ContentHandler handler = new BodyContentHandler(-1);
```

The problem was solved and we could get the actual size of the documents as we can see below.

**Table 2: Original vs Parsed Document Size Analysis**

Document Name	Original size(KB)	After parsing(KB)
ABCofABCD.pdf	3,245	193
Big_Java.pdf	13,325	2,335
DB System concepts.pdf	209,416	2,818

Still when the heap-space and the size truncation problems were solved, there was another problem we encountered. Without giving any error message, the speed of uploading a big document was still slow. This problem was caused by creating a lot of debugging log-messages and this was not really a noticeable problem to some extent when the file size is small or medium, but when we tested loading big documents, still the speed remained unacceptable. This was solved by setting the log level of the root logger to INFO, a much lower level, as follows:

```
public static Logger logger = null;
static {
    logger = Logger.getRootLogger();
    logger.setLevel(Level.INFO);
}
}
```

All problems mentioned above were solved and finally one could load a document of any size within a few seconds without any problem. After uploading a document successfully, the next job was to save the uploaded document in J-ISIS record and to index the loaded content. In this case also, for our purpose we needed to test files ranging from small size (e.g. article with about 10 pages) up to big documents (e.g. e-books with hundreds of pages). When we tried to save small size files and indexing it was not a problem. But, when we increased the file size we couldn't save it due to "ConnectionNIO::sendMessage Exceptionjava.io.UTFDataFormatException" problem.

The above problem was caused by the fact that J-ISIS is a Client-Server application and when we started the J-ISIS database server on the local machine, all database operations are done by means of establishing the communication and transferring different messages between the client and the server through TCP/IP. The org.unesco.J-ISIS.corelib.common.ConnectionNIO class is responsible of sending messages from the client to the server and vice versa.

The original method used to encode the document content into a record and to read the content was the following:

```
private void writeObject(java.io.ObjectOutputStream out) throws IOException {
    out.defaultWriteObject();
    out.writeUTF(value_);
}

private void readObject(java.io.ObjectInputStream in)
    throws IOException, ClassNotFoundException {
    in.defaultReadObject();
    value_ = in.readUTF();
}
}
```

The "UTFDataFormatException exception" is thrown by ObjectOutputStream.writeUTF method in the Java API if a string attribute is greater than 64KB in length (see:

[http://bugs.sun.com/bugdatabase/view\\_bug.do;jsessionid=f175054d850bffffff99fab937e1381a2?bug\\_id=4039553](http://bugs.sun.com/bugdatabase/view_bug.do;jsessionid=f175054d850bffffff99fab937e1381a2?bug_id=4039553) ). This was a bug into the original method of Java. To overcome this problem, the above writeObject and readObject methods were changed as follows:

```
private void writeObject(java.io.ObjectOutputStream out) throws IOException{

    byte[] buf = value_.getBytes("utf-8");
    out.writeInt(buf.length);
    out.write(buf);

}

private void readObject(java.io.ObjectInputStream in)
    throws IOException, ClassNotFoundException {

    int len = in.readInt();
    byte[] buf = new byte[len];
    in.readFully(buf);
    value_ = new String(buf,"utf-8");
}
}
```

As was illustrated above, we experienced that programming for a database with ‘normal’ and typical values is not the same as programming for (very) large document contents. Several solutions to overcome problems by Java and pre-existing classes of J-ISIS had to be provided, but the final result now available seems to work fine and fast also with very large documents.

### 5.2.3. Testing J-ISIS Performance

In this section, results of testing J-ISIS performance in general and in comparison to the well known digital library software Greenstone will be presented.

To assess the performance of J-ISIS in terms of storage space, indexing and searching speed and capability of handling different file formats, different tests have been undertaken by creating a digital library collection of 200 documents with different file formats, mostly PDF (as can typically be expected) and a few .doc(x), .html, .ppt and .xls files. The average size of the original documents is 1940KB with a maximum of 209MB (a big PDF) and typically the smallest files being .html at about 20KB. The average size of the extracted text files is 106KB with the maximum being 2818KB and the smallest being 1.3KB. We found a Pearson correlation coefficient of 0.56 in between the two series of respectively original and extracted sizes, indicating a large variety of compression due to different content types of the files.

As a result, J-ISIS has been found to be quite powerful software in terms of the above mentioned parameters. For instance, it only took 3 seconds for indexing one document on average, it can store any file size, searching and viewing of results is almost immediate. All documents were included in the collection without exception.

Moreover, to compare the performance of J-ISIS with Greenstone, Greenstone software has been installed and the same collection (with 200 documents with different formats) was used to be consistent and allow comparison with J-ISIS. In case of Greenstone, it has reported 189 documents were successfully processed, 1 was unrecognized and 11 were rejected, despite having used the most recent version (v2.85, published fall 2011) with latest PDFBox installed as per the instructions. So, this shows us J-ISIS has more powerful mechanisms in extracting and recognizing the file formats as a result of using Tika, compared to Greenstone which uses different plug-ins for handling different documents.

In addition, we have compared the indexing speed and storage space of J-ISIS with Greenstone. All tests were done on a Windows PC with DualCore 2,20Ghz CPU and 3 GB of RAM. Even if Greenstone has its own indexing engine (MGPP), it also uses Lucene to support incremental indexing and for comparison matters, Lucene has been used as indexer. As a result, Greenstone took 52 seconds to index 189 files whereas J-ISIS took only 12,5 seconds for the whole set of 200 documents.

Finally, the comparison with storage space was undertaken and for storing these 189 documents, Greenstone used 626 MB, whereas J-ISIS only used 204 MB for 200 files. This is huge difference with consequences for librarians planning to do bigger collections: for only 200 documents already the storage efficiency drop of Greenstone costs hundreds of MB's. Extrapolating this for thousands of documents, one can see that Greenstone needs much more storage space, especially because the difference is not only in the indexes (J-ISIS taking only 1/10<sup>th</sup> of the Greenstone index storage space, where again for some reason for each document a copy is stored and a separate XML-file is to be kept) but also in storage of the documents themselves, meaning that the storage space will grow monotonically with the number of documents.

While doing the tests, we have observed that Greenstone is capable of batch importing, while J-ISIS is not, and this makes the initial collection creation faster, whereas in case of J-ISIS it has to be done one by one, taking more time for this initial creation of the collection. When

doing incremental building of the collection however, the J-ISIS approach, which takes only seconds to add a document, is much faster.

A summary of the findings is presented in the following table:

**Table 3: Comparison of Greenstone Digital Library System with J-ISIS**

Feature/Criterion	Greenstone Digital Library System	J-ISIS	Comments
Success coping with test collection documents	91%	100%	GSDL: v2.85 with new PDFBox installed as instructed
Total time to build collection	15:45	33 min. (10 seconds average per document)	GSDL : batch import possible, J-ISIS : speed highly dependent on file-selection interaction both : all docs in 1 folder available
Average time adding one document	6 min	10 sec	GSDL : minimal rebuild only, no hashing for optimal speed, plugins list optimized to collection profile both : no meta-data added but embedded meta-data processed
Indexing full collection	55 sec	9 sec	Lucene is used in both cases
Total storage for collection	626 MB	204 MB	GSDL : keeps copy of original files in 2 locations
Space occupied by indexes	223 MB	23 MB	
Web-interface end-users	Yes	Yes	J-ISIS : Web-JISIS prototype only
Possibility to edit text-contents of document	No	Yes	e.g. tagging inside document
Advanced features e.g. page-browsing, intra-document sections	Yes	No	GSDL, as a dedicated DL software, emphasizes these features but they are not used often

As can be seen, both softwares are quite different in their architecture. E.g. GSDL allows batch-import, which could be handy in specific circumstances. GSDL uses the file-system as the 'database' manager and organizes the collections therefore in a complex set of folders with many subfolders. This is not as fast as a powerful database like Berkeley DB can deal with records, and this is a significant advantage of the database-based approach of J-ISIS.

The main difference might be in the storage space needed for typical collections, where the observed major differences will be magnified in view of the habit of GSDL to not copy the original files only in the 'import' folder but again in the 'archives' and even index folders.

On the other hand, Greenstone will allow some advanced features to be applied, especially with regards to browsing the database, not only by 'classifiers' (e.g. metadata-fields) but also by page and document-subsections. This makes GSDL the excellent solution it is, without doubt, for digital libraries. But all this clearly comes at a speed and storage price.

To sum up, even if we take into account that a full comparison needs probably a more elaborated set-up and discussion (in view of the many architectural differences), we have discovered that J-ISIS outperforms Greenstone in terms of storage space and indexing speed. This is a remarkable result in view of Greenstone having been designed with digital library capability in mind, whereas J-ISIS was used for traditional library database usage.

### **5.3.     *The OAI-PMH***

OAI-PMH is another important requirement for normal and digital library applications. It enhances the interoperability between different applications by means of harvesting the metadata and contents stored in these applications.

The advantage of implementing OAI-PMH for digital library software like J-ISIS is the development and promotion of interoperability standards that aim to facilitate the dissemination of content (Hornik, 2009). This means that the use of the OAI-PMH feature with J-ISIS will allow non-J-ISIS users to access to metadata used in J-ISIS and access the content within J-ISIS. This feature is quite interesting and enhances the interoperability of applications. In my view this feature has to be implemented in the frame of the whole J-ISIS, not only to J-ISIS digital library technology. We leave this as a task to do for the feature with

J-ISIS due to the fact that its scope is a bit broad and should be implemented in the overall J-ISIS frame, for all J-ISIS application types.

#### **5.4. Interactivity**

Interactivity is also found to be another important requirement to digital library software in the new 'Web 2.0' tradition. Interactivity is used to co-operate people/scholars by enabling them to comment/give their opinion in the published library resources. Therefore, users are not only intended to be passive users by only reading the published content but they could give their feedback. Even in traditional libraries reader-evenings are organized to allow readers and users to discuss their reading and exchange their views, which is the 'social networking' in non-digital context, which can be coped with nowadays rather well by modern web-techniques.

As J-ISIS is a database-based application, it will be rather easy to provide some extra (repeatable) fields for user-comments and contributions, and a user-interface for editing these via the web-application (Web-JISIS). This could be another nice feature to be implemented in the framework of another project.

## **Chapter Six: Conclusions and Recommendations**

Digital library technology has changed the way information is collected and accessed in library systems. This plays a vital role in view of information availability, dissemination and accessibility. In view of this, this study aims at testing the capability of UNESCO's software J-ISIS, which was developed under Free and Open Source Software license, with regards to the digital library concept and enhancing the capability of J-ISIS by adding new digital library functionality.

Digital libraries have their own requirements which need to be fulfilled in order to accomplish their role and offer the required service to the end user. This includes the capabilities to deal with a diversity of metadata, full-text indexing, diversity of document formats, Unicode, OAI-PMH, and interactivity.

As a classic ISIS-technology, the ABCD library software which uses PHP as programming language and ISIS database for storage of data was enabled for digital library functions unlike its traditional library automation of bibliographic data following my attachment work on the software. This means it can store the physical content of the resource by giving access to the text-contents of the original document up to a size of 32KB (with ISIS old version) and 1MB (with ISIS extended version), along with an automatically created hyperlink to the original file format. However, due to limitations associated with the classic ISIS-technology, such as size limitations and not being Unicode compatible, it cannot be fully claimed that ABCD can be used for digital library applications. Therefore, there was a need to solve the limitations of ABCD as a digital library software. This has motivated me to search a better solution using the latest version of ISIS (J-ISIS) which is the core aim of my thesis.

J-ISIS is a Java based client-server implementation of ISIS and uses the TCP/IP protocol to transfer messages from client to server and vice versa. J-ISIS is enriched and uses relatively new technologies such as Berkeley DB for storage, Lucene for full-text indexing and retrieval, and as a result of my work, Tika for extracting text from different document formats. The technology embedded with J-ISIS makes it a powerful software with respect to providing unlimited storage capability, the use of many scripts (Unicode compatibility), full-text indexing and handling of documents with different formats. Thus, in principle, J-ISIS is a good candidate to be used for digital library applications as it fulfills the fundamental needs

of digital libraries. However, there was not a practical test regarding the existing features and adding some new ones to make it capable of performing digital library functions.

The only originally available approach of copy-pasting content into a record was unacceptable practically in view of its time consumption and being a tedious process. There was a need to implement a new feature and for this purpose it was needed to create the interface elements. As a result, files can now be uploaded into J-ISIS worksheet in which the new 'digital library' field-types have been defined.

Following the different tests that I have conducted with J-ISIS, it has been proven that J-ISIS indeed can do more than traditional library applications (e.g. bibliographic based library automation) and it is indeed a digital library capable software as it provides high storage capability, Unicode compatibility, full-text indexing, and can handle a diversity of metadata and a diversity of document formats. Moreover, the general performance of J-ISIS has been tested and it has found to be quite impressive as J-ISIS can store any document size except the limits of the available physical system, it can handle documents written in different scripts, can handle any document format and provide a web based access to its collections by means of Web-JISIS. In addition, it has a very fast indexing and searching capability which can be expressed as a matter of seconds even if the document being indexed and searched is large (e.g. indexing a collection of 200 files took only 9 seconds).

This paper has identified some additional features which are very important to make J-ISIS fully digital library capable software such as OAI-PHM and interactivity. Even though, the OAI-MHP is a very important feature to enhance interoperability with other applications, due to the fact that this has to be designed and implemented as part of the whole J-ISIS framework, including traditional and digital library, it needs a lot of preparation and discussion with the experts. As a result, this was left out of the scope of this thesis, and we recommend it to be in the list of to-do activities in further J-ISIS development stages. In addition to this, I recommend interactivity to be included as another digital library requirement building on the flexible database-architecture of J-ISIS and the Web2.0 facilities of the JSP and Struts-technology used in the WWW-interface.

To sum up, I personally believe that this paper has solved a lot of bottlenecks with J-ISIS and also confirms and makes J-ISIS a digital library capable software by implementing a new feature for easy incorporation of a variety of document formats.

I strongly recommend UNESCO ( the owner of the software), in view of its high interest in digital libraries and Cultural Heritage preservation, to invest more and add other digital library features as its benefit is far- reaching as proved in this thesis with some of its features and capabilities more specifically we recommend to work on :

- The elaboration of the prototype Web-JISIS interface to make it fully functioning, e.g in the presentation of search results and adding the ‘edit’ feature, which will also allow providing interactivity for end-users;
- Giving some input into J-ISIS to polish it and add some remaining smaller interface functionality like system parameter settings and OAI functionality;
- Doing more promotion on J-ISIS as it can provide not only a means for management of bibliographic databases (and library automation) but also of digital library collections in an integrated way, so that smaller information centers can avoid having to duplicate efforts over different softwares.
- Improving online availability of resources on ISIS in general and J-ISIS more specifically. While working with J-ISIS, the main problems that I have encountered are problems related to material availability for consultation due to the fact that ISIS is old software and the users are being in a low technical level. Moreover, I couldn’t find published materials about J-ISIS and Web-JISIS as they are at their initial development and maintenance stage. This leads to investment of more time than of anticipated.

As a result of a lot of efforts and time investment in UNESCO’s J-ISIS software, it can deal with digital library. More investment and attention towards J-ISIS is quite important to benefit fully from the software in view of the already available features and capabilities, now also for digital libraries.

## References

- Abboy, I., Hoskins, R. (2008): *The use of CDS/ISIS software in Africa*, Innovation: A Journal for Appropriate Librarianship, 36 (June) 17-37.
- Apache (2011) : Apache Lucene-Features, <http://lucene.apache.org/java/docs/features.pdf>
- Arms, W.Y. (1995): *Key Concepts in the Architecture of the Digital Library*, D-Lib Magazine, Reston, Virginia.
- Barna, P., Frasincar, F., Houben, G., Vdovjak, R. (2003): *Methodologies for Web Information System Design*, Proceedings of the International Conference on Information Technology: Computers and Communications (ITCC.03). ISBN: 0-7695-1916-4.
- Bartunov, O., Sigaev, T., (2007): *Full-text Search in PostgreSQL: A Gentle Introduction*, Moscow, Russia.
- Berhe H.H. (2011): *Extending the Integrated Library software (ABCD) with Digital Library function*, paper submitted for Applied Computer Sciences M.Sc. attachment, VUB, Brussels.
- BIREME(2007), *Basic concepts of CDS/ISIS databases: an introduction to the use of CISIS*, Sao Paulo, BIREME / PAHO / WHO, [bvsmodelo.bvsalud.org/download/cisis/CISIS-ConceptosBasicos-en.pdf](http://bvsmodelo.bvsalud.org/download/cisis/CISIS-ConceptosBasicos-en.pdf).
- BIREME (2008): *Rio Declaration on the Future of the ISIS Software*, ISIS3WC: World Congress of CDS/ISIS Users: Informative Report, BIREME, Rio de Janeiro, Brazil, <http://www.eventos.bvsalud.org/agendas/isis3/public/documents/ISIS3%20report-140102.pdf>.
- Brust, A.J. (2011): *NoSQL and the Windows Azure platform*, investigation of an Unlikely Combination, Blue Badge Insights, Inc.
- Buxton, A. (2002): *The WWWISIS handbook (for versions 4 and 5)*. Sao Paulo: BIREME.
- Buxton, A. (2010): *PUTTING YOUR CDS/ISIS DATABASE ON THE INTERNET*, Institute of Development Studies, Brighton, UK.
- Candela, L., Castelli, D., Pagano, P., Thanos, C., Ioannidis, Y., Koutrika, G., Ross, S., Schek, H., Schuldt, H. (2007): *Setting the Foundation of Digital Libraries*, D-Lib Magazine, ISSN: 1082-9873, Volume 13 Number ¾ .
- Carpenter, B. (2011): *Lucene Version 3.0 Tutorial*, LingPipe, Inc, Excerpted from: Text Analysis with LingPipe 4.0, <http://alias-i.com/lingpipe-book/index.html>.
- Carswell, J.D (2001): *Deploying integrated web-based spatial applications within an Oracle database environment*, Digital Media Centre, Conference papers, Dublin Institute of Technology, Dublin, Republic of Ireland.
- Comstock, S. (2006): *An Introduction to Unicode*, <http://www.trainersfriend.com/Papers/uncdtalk.pdf>.
- Couchbase (2011): *NoSQL Database Technology*, Post-relational data management for interactive software systems, <http://www.couchbase.com/sites/default/files/uploads/all/whitepapers/NoSQL-Whitepaper.pdf>.
- Dauphin JC. (2003) : *ISIS today and tomorrow*, paper presented at the ISIS Experts Meeting in Eschborn, 3-4 April 2003, version 1.3, UNESCO, Paris, France.
- Dauphin, JC. (2008): *ISIS3WC: J-ISIS*, World Congress of CDS/ISIS Users: Informative Report, BIREME, Rio de Janeiro, Brazil, <http://www.eventos.bvsalud.org/agendas/isis3/public/documents/ISIS3%20report-140102.pdf>.

- Dauphin, JC. (2010a): *J-ISIS Quick Tutorial*, <http://www.kenai.com/downloads/j-isis/J-ISIS Quick Tutorial October 2011.pdf>.
- Dauphin, JC. (2010b): *web J-ISIS reference manual*, <http://www.kenai.com/downloads/j-isis/Web-JISIS.pdf>.
- Dempsey, L. (2006). *The (Digital) Library Environment: Ten Years After*, Ariadne, Issue 46, February, <http://www.ariadne.ac.uk/issue46/dempsey/>.
- De Smet, E. (1999): *The ISIS-software family: an introduction for novice/prospective users*, University of Antwerp, Belgium.
- De Smet, E (2008a): *ABCD, ISIS3WC: World Congress of CDS/ISIS Users: Informative Report*, BIREME, Rio de Janeiro, Brazil, <http://www.eventos.bvsalud.org/agendas/isis3/public/documents/ISIS3%20report-140102.pdf>.
- De Smet, E. (2008b): *The ISIS-software family: from 'Free and Open' to 'Free and Open Source Software'*, journal of appropriate librarianship and information work.
- De Smet, E (2009): *The abc of ABCD: the Reference Manual Version 1.0*, [http://bvsmodelo.bvsalud.org/download/abcd/manuais/ABCofABCD\\_atualizado.pdf](http://bvsmodelo.bvsalud.org/download/abcd/manuais/ABCofABCD_atualizado.pdf).
- De Smet, E. (2010): *Some ISIS-software history and technical background to the new FOSS integrated library system 'ABCD'*. Liber quarterly: the journal of European Research Libraries / European Research Libraries - ISSN 1435-5205 - 19:3/4, p. 324-335.
- De Smet, E. (2011): *ABCD as a digital library tool*, Paper presented as 'Invited Speaker' at the International Conference on Digital Libraries and Knowledge Organization, Delhi.
- FAO and UNESCO (2005): *Information Management Resource Kit, Module on Digitization and Digital Libraries*, [http://www.nlv.gov.vn/nlv/index2.php?option=com\\_docman&task=doc\\_view&gid=283&Itemid=82](http://www.nlv.gov.vn/nlv/index2.php?option=com_docman&task=doc_view&gid=283&Itemid=82)
- Grosso, W. (2004): *Berkeley DB, Java Edition I: The Basics*, the source for Java technology collaboration, Java.net.
- Hahmann, T. (2008): *Apache and CGI*, CSC 309 Tutorial, [http://www.cs.toronto.edu/~torsten/THahmann\\_CSC309\\_Tutorial1\\_ApacheCGI.pdf](http://www.cs.toronto.edu/~torsten/THahmann_CSC309_Tutorial1_ApacheCGI.pdf).
- Haile, G., de Smet, E., Mulugeta, Y. (2010): *Challenges of Traditional Methods of Preserving Ethiopian Ge'ez Manuscripts and Promises of Digital Preservation in Tigray*, paper presented at the International Conference on 'Preservation of African Manuscripts', Addis-Abeba.
- Hansen, C. (2005): *Unicode*. [http://www.findthatfile.com/search-6210321-hPDF/download-documents-data\\_01.pdf.htm](http://www.findthatfile.com/search-6210321-hPDF/download-documents-data_01.pdf.htm).
- Hatcher, E. (2004): *Lucene Intro*, <http://today.java.net/pub/a/today/2003/07/30/LuceneIntro.html>.
- Hopkinson, A. (1996): *CDS/ISIS: UNESCO's information retrieval package for microcomputers and the VAX minicomputer*. Encyclopedia of library and information science 57(20): 72-88.
- Hopkinson, A. (2008): *CDS/ISIS INFORMATION*, *Information Development* 24: 258-262
- Hopkinson, A. (2010): *Information Development*, <http://idv.sagepub.com/content/26/4/260.citation> SAGE.
- Hornik, K. (2009): *Metadata Harvesting with R and OAI-PMH*, <http://cran.r-project.org/web/packages/OAIHarvester/vignettes/oaih.pdf>.

- Jayakanth, F. (2001): *Implementing WWWISIS for providing web access to bibliographic databases*. Inspel 35: 42-58.
- Kassim, A.R.C., Kochtanek, T.R. (2003): *Designing, implementing, and evaluating an educational Digital Library resource*, Online Information Review, Vol. 27 Iss: 3, pp.160 – 168.
- Keuchler, B. and Vaishnavi V. (2011): *'Promoting Relevance in IS Research: An Informing System for Design Science Research'*, Informing Science: the International Journal of an Emerging Transdiscipline Volume 14, 2011, p. 125-138, <http://www.inform.nu/Articles/Vol14/ISJv14p125-138Kuechler570.pdf>.
- Lagoze, C., Krafft, D.B., Payette, S., Jesuroga, S. (2005): *What is a Digital Library Anymore, Anyway?* D-Lib Magazine, Volume 11 Number 11.
- Lai, Eric (2009): *No to SQL? Anti-Database Movement Gains Steam*, Computerworld Software, July 1, [http://www.computerworld.com/s/article/9135086/No\\_to\\_SQL\\_Anti\\_database\\_movement\\_gains\\_steam](http://www.computerworld.com/s/article/9135086/No_to_SQL_Anti_database_movement_gains_steam).
- Lee, G.T., Dahlan, N., Ramayah, T., Karia, N., Asaari, M.H.A.H. (2005): *Impact of Interface Characteristics on Digital Libraries Usage*, Malaysian online Journal of Instructional Technology, ISSN: 1823-1144, Volume 2, No. 1. Malaysia.
- Mahmood, K. (1998): *Use of Micro CDS/ISIS in Pakistan: a survey*. Inspel 32: 23-39.
- Matovelo, D., de Smet, E. (2005): *CDS/ISIS software for libraries. Information and Documentation 25*. Wageningen (the Netherlands): Technical Centre for Agricultural and Rural Cooperation.
- Marbi (1996): *The MARC 21 Formats: Background and Principles*, American Library Association's ALCTS/LITA/RUSA Machine-Readable Bibliographic Information Committee in conjunction with Network Development and MARC Standards Office Library of Congress.
- Meyer, E., Grussenmeyer, P., Perrin, J.-P., Durand, A., Drap, P. (2007): *A web information system for the management and the dissemination of Cultural Heritage data*, Journal of Cultural Heritage, vol. 8, 4, pp. 396-411.
- Mishra, R.K. and Mishra M.K. (2006): *Creation of museum database using WINISIS: a case study*. In Reddy, M.S., Babu, T.A. and Ramaiah, L.S. (eds.) *Developing cyber libraries*. Mumbai (India): Allied Publishers. pp. 376-403.
- McCray, A.T., Gallagher M.E. (2001): *Principles for Digital Library Development*, COMMUNICATIONS OF THE ACM May 2001/Vol. 44, No. 5, New York, NY, USA.
- NISO (2001): *Understanding Metadata*, <http://www.niso.org/publications/press/UnderstandingMetadata.pdf>.
- NoSQL (2009): *Wikipedia*, <http://en.wikipedia.org/wiki/NoSQL>.
- Olson, A.M., Bostic, K., Seltzer, M. (1999): *Berkeley DB*, Proceedings of the FREENIX Track: USENIX Annual Technical Conference, Monterey, California, USA.
- Oracle (2011): *Hadoop and NoSQL Technologies and the Oracle Database*, An Oracle white paper, <http://www.oracle.com/technetwork/database/hadoop-nosql-oracle-twp-398488.pdf>.
- Prasad, A. R. D., Patel, D. (2005): *Lucene Search Engine – an overview*, DRTC-HP International Workshop on Building Digital Libraries using Dspace held on 7th – 11th March, at DRTC, Bangalore.
- Ramalho, L.G. (2011): *From ISIS to CouchDB: Databases and Data Models for Bibliographic Records*, Code {4} lib JOURNAL, ISSN 1940-5758, see <http://journal.code4lib.org/articles/4893>.

- Reis, D., Freire, N. (2008): *OAI-PMH implementation and tools guidelines*, [http://www.theeuropeanlibrary.org/portal/organisation/handbook/Documents3/TELpl us\\_D2.1\\_31052008.pdf](http://www.theeuropeanlibrary.org/portal/organisation/handbook/Documents3/TELpl us_D2.1_31052008.pdf).
- Rodriguez, K. (1995): *CDS/ISIS: a statistical analysis of usage in Latin America and the Caribbean*. International information and library review 27: 225-235.
- Sathish, K., Maly, K., Zubair, M., Lie, X. (): *RVOT (2009): A Tool for Making Collections OAI-PMH Complaint*, Russian Conference on Digital Libraries, <http://rvot.sourceforge.net/documents/RapidVisualOAITool.pdf>.
- Seeley, Y (2007): *Full-text Search with Lucene*, Amsterdam, Netherlands.
- Singh, S. (2003): *Digital Library: Definition to Implementation*, Lecture Delivered at Ranganathan Research circle, Delhi.
- Su, D.C. (2002): *Performance Analysis and Optimization on Lucene*, <http://www.stanford.edu/class/cs276a/projects/reports/dsu800.pdf>.
- Tenopir, C. (2003): *Use and Users of Electronic Library Resources: An overview and Analysis of Recent Research Studies*, (With the assistance of Brenda Hitchcock and Ashley Pillow), Council on Library and Information Resources Washington, D.C.
- The ISIS3 Conference (2010): *New Challenges for a new future of ISIS*, [http://www.eventos.bvsalud.org/agendas/isis3/public/documents/01\\_ISIS\\_challenges\\_ppt-175255.pdf](http://www.eventos.bvsalud.org/agendas/isis3/public/documents/01_ISIS_challenges_ppt-175255.pdf).
- Tikhonov, O., Mattman, C. (2010): *Understanding information content with Apache Tika*, <http://www.ibm.com/developerworks/opensource/tutorials/os-apache-tika/os-apache-tika-pdf.pdf>.
- Tweed, R. and James, G. (2010): *A Universal NoSQL Engine, Using a Tried and Tested Technology*, Creative Commons Attribution CC-BY 3.0.
- UNESCO (1989): *Mini-micro CDS/ISIS reference manual (version 2.3)*. Paris, France.
- UNESCO (2004): *CDS/ISIS for Windows, reference manual (version 1.5)*, p.87, Paris, France.
- Vijayakumar, J.K., Jeevan, V.K.J. (2001): *Digital Library Development: Major Issues of Externally Puplicated Contents*, INFLIBNET Center, Ahmedabad, India.
- Web-J-ISIS Reference Manual (2010): <http://kenai.com/downloads/j-isis/Web-J-ISIS.pdf>.
- Wikipedia (2011): *Information Retrieval*, [http://en.wikipedia.org/wiki/Information\\_retrieval](http://en.wikipedia.org/wiki/Information_retrieval).
- Witten, I.H., Nevill-Manning, C., McNab, R., Cunningham, S.J. (1998): *A Public Library Based on Full-text Retrieval*, *Communications of the ACM*, Volume 41 Issue 4, April 1998, New York, NY, USA.
- Witten, I.H., Bainbridge, D., Boddie, S.J. (2001): *Greenstone: Open-source Digital Library Software*, *D-Lib Magazine*, volume 7 Number 10, ISSN 1082-9873.
- Witten, I.H., Bainbridge, D., Paynter, G., Boddie, S. (2002): *Importing Documents and Metadata into Digital Libraries: Requirements Analysis and an Extensible Architecture*, *Research and Advanced Technology for Digital Libraries*.