MAKING LINKS

Wendy Hall and Susan Malaika†*
* Department of Electronics and Computer Science , University of Southampton, UK
† IBM Santa Teresa Laboratory, San Jose, California, US

Abstract
This paper describes the major issues in hypermedia systems (systems that manage links in unstructured data) and in relational database systems (systems that manage links in structured data). Particular attention is given to the ways that linkages and relationships between data items are handled both statically, prior to a link being used, and dynamically at the time a link is required. The effect of the World Wide Web and Java on both hypermedia and relational database systems is described. The paper then outlines future directions in hypermedia and relational database systems, including improved integration of structured and unstructured data.

1 MAKING LINKS IN HYPERMEDIA SYSTEMS

1.1 In the beginning

We all think by making associations between different pieces of information stored in the seemingly infinite database of our brains. Sometimes we don't even remember how we originally made these links but they enable us to retrieve both related and seemingly unrelated pieces of information. Ever since people have been able to read and write, they have wanted to be able to capture such cross-references across the written word. The example that would be most well known to all of us is the amazing amount of scholarly work represented in cross-referencing the ancient scriptures such as the Bible and the Talmud.

The seventeenth century polymath, Samuel Hartlib, recognised the implicit cross-referencing structure of an encyclopaedia. He talked about how fascinated he was by the interconnectedness of everything in the Universe. To quote from his writings:

> "The most exact Encyclopaedias which I could ever lay my hands on, seemed to me like a chaine, neatly framed of many links.... this living tree, with living roots, and living fruits of all the Arts and Sciences, I meane a Pansophy which is a lively image of the Universe .... every where covering it selfe with fruit" [1]

Which just goes to show that nothing is ever really new. Hartlib was talking about links between associated pieces of information over 400 years ago.

1.2 The Early Visionaries

In the twentieth century, the prospect of digital computers to help us make and use such cross-references became apparent. One of the early visionaries in this field was Vannevar Bush, who was the scientific advisor to Roosevelt in the second world war. He worked on the Manhatten project and foresaw the explosion of scientific information which makes it impossible even for specialists to follow developments in a single discipline, let alone the multitude of disciplines involved in a large scientific or engineering project.

In his seminal paper “As We May Think”, which was published in the Atlantic Monthly in 1945 [2], Bush proposed the Memex (or memory extender) system which he described as a sort of mechanized private file and library. His ideas were based on a mechanical machine using microfilm or something similar, but they are obviously even more applicable to the world of digital computers. Bush described the Memex in theory and it was never implemented in its original form, but the ideas live on in today’s Internet society.

His design, incorporated the idea of associative indexing, or “As we may think”, and the building and following of “trails” of information. The Memex had a scanner that would scan in all information of interest to the user. Here we have the concept of a machine that photographs everything the scientist looks at and stores for future use. This may sound like the realms of science fiction, but people are just beginning to use such devices in everyday life today.

Bush even forecast a new breed of information scientists called “trail blazers” whose job it would be to create useful or relevant trails through vast information spaces. Bush likens the idea of creating a trail to that of cross-referencing or linking. To quote from his paper:-

> “The Memex affords an immediate step to associative indexing, the basic idea of which is a provision whereby an item may be caused at will to select another immediately and automatically. This is the essential feature of the Memex. The process of tying two items together is the important thing.”

The idea of making and following links between pieces of digital information is now known as *hypertext* and although Bush never really returned to the idea after the publication of “As We May Think”, he is known as the “grandfather of hypertext” because of the inspiration that his paper gave to others who followed in his footsteps. Here is another quote from Bush:-

> “Wholly new forms of encyclopaedias will appear, ready made with a mesh of associative trails running through them, ready to be dropped into the Memex. The lawyer has at his touch the associated opinions and decisions of his whole experience and of the experience of friends and authorities. .... The physician, puzzled by a patient’s reactions, strikes the trail established in studying an earlier similar case, and runs rapidly through analogous case histories, with side references to the classics for the pertinent anatomy and histology. The chemist, struggling with the synthesis of an organic compound, has all the chemical literature before him in his laboratory, with trails following the analogies of compounds, and side trails to their physical and chemical behaviours.”

These ideas are as relevant today as they were in 1945 - maybe even more so because now we can see that the implementation of such a system is actually possible, if still extremely complex.

The name of Douglas Engelbart should be known by everybody who has ever used or even seen a modern computer. But he is one of those unsung hero’s of scientific endeavour. Engelbart is credited with being the inventor of word processing, screen windows and the mouse. In other words all the interface tools that made the computer something that could be used outside of the scientific laboratory.

In the early 60’s Engelbart realised that computers were capable of dealing with much more than just scientific data. He became interested in developing ways to provide systems and interfaces that would help what he called “knowledge workers” in their everyday working environment. He began the Augment project, otherwise known as the NLS, or oN-Line System, in 1962, its main objective being to develop a system to *augment* human capabilities and support group working [3].

NLS had several hypertext features, although at the time they weren’t described as such. But its most important feature was the concept of interacting with the information on the screen using a point and click device, today known as a mouse. Engelbart gave a live demonstration of his system at a very large computing conference in San Francisco in 1968. To the people in the audience at the time it must have seemed like something out of science fiction. To quote from Jakob Nielsen's classic book on hypertext:

> “In spite of the successful demo of NLS, the US government dropped its research support of Engelbart in 1975 at a time when he had more or less invented half the concepts of modern computing. After the Augment project was as good as terminated, several people from Engelbart’s staff went on to Xerox PARC and helped invent many of the second half of the concepts of modern computing.” [4]

Engelbart is still actively pursuing his original augmentation and shared working ideas through the “Bootstrap Institute” based at Stanford.

Ted Nelson describes himself as the father of hypertext, and quite rightly so. It was in fact Nelson who coined the terms hypertext and hypermedia in 1965. He readily admits he was inspired in his ideas by Bush and Engelbart and the potential interconnectedness of all information.

Nelson views hypertext as “non-sequential writing”. He believes that no digital information ever created should be thrown away. It should be available for re-use and referencing using a powerful system of hypertext linking to maintain the authorship of every piece of information and give credit of authorship to the originator of every bit and byte in the system. Very early on in his work he developed the concept of Xanadu - a repository for everything that anybody has ever written - a truly universal hypertext , which is fully described in his book Literary Machines, published in 1981 [5]. His vision extended to the idea of high-street Xanadu "kiosks" where anyone could get access to the universal knowledge repository

Xanadu has never (yet) been implemented in full although parts have been. For a while it was “owned” by Autodesk, but the Intellectual Property Rights are now back with Nelson who is based in Japan working on Xanadu and focusing amongst other things on system support for pay per byte copyright mechanisms.

1.3 Pioneering Systems

So now we progress to the 1970's when computer technology had developed to the point where it was technically possible to try building hypertext systems. The earliest versions ran on mainframe computers with command line interfaces, but they did allow researchers to experiment with the practicalities of automating the process of making links.

Credit for the first purpose-built hypertext system usually goes to Professor Andy van Dam at Brown University in Providence, USA for the mainframe based FRESS system [6]. As the increasingly interactive workstation computers evolved in the 1980's, so the hypertext systems that were being developed could have more sophisticated interfaces. Knowledge Management System [7], which evolved from the strangely named ZOG system developed at Carnegie Mellon, and Hyperties [8] from the University of Maryland, were both essentially frame based hypertext systems, which allowed screenfuls of information to be displayed one at a time, with various methods deployed to enable users to follow links from one frame to another. Notecards [9] from Xerox PARC, was based on scrolling text windows and was designed to promote group work. All these systems grew out of research projects and were eventually released commercially with varying degrees of success.

Special mention must be made here to the Intermedia system [10], which was developed at Brown in the 1980's. The intention of the Intermedia team was to create a model for hypermedia functionality handling at the system level, where linking would be available to all participating applications, with the information about links stored in a database. The project was highly innovative and very much ahead of its time. It also pioneered the use of hypertext on a large scale in education through the work of George Landow and others. However, because of bad decisions and bad luck about the development environment, Intermedia never became a commercial product.

1.4 Hypermedia Systems for the Personal Computer

By the middle of the 1980's the world was becoming accustomed to the idea of the personal computer. This enabled the development of a new generation of hypertext systems which finally brought hypertext and hypermedia, if not to the masses, then certainly to the attention of a sizeable proportion of personal computer users.

The first hypertext system for personal computers was Guide, which was released by a Scottish company, Office Workstations Ltd, in 1986. It was based on the research work of Professor Peter Brown at the University of Kent [11] and used a scrolling text windows to display the information in the system to the user. I liken it to a folding editor. It has a fixed set of link types - pop-up, reference, inline replacement - and maintains a hierarchical view of the hyperdocument structure to help users navigate around the information without getting disoriented.

Guide was the first popular hypertext system, and has been applied very successfully to many large-scale industrial applications. It also had an edge in that it was the first cross-platform system with both IBM pc and Macintosh versions. However, the success of the Macintosh version was severely curtailed by Apple's release of HyperCard in 1987 [12].

HyperCard was not designed as a hypertext system, it was more a graphics prototyping environment, but it allowed cards of information to be easily linked together. Apple renamed the product, previously called WildCard, just prior to its release. The first international hypertext conference was held in the summer of 1987 so Apple had cleverly spotted a marketing opportunity and gave HyperCard away with every Macintosh sold.

The model for information management in HyperCard is that of a stack of cards. The words that are high-lighted in the text are called buttons. If you click on one of these buttons, you activate a link that takes you to another card in the application which usually tells you more about whatever was indicated by the button. The look and feel of the cards, and their functionality are completely programmer definable. HyperCard spawned several similar products such as SuperCard, Toolbook, Spinnaker Plus etc. It was very easy to build simple stacks but it positively encouraged the use of “goto” linking and spaghetti hypertexts! For computer scientists this should ring bells in terms of the programming language debate of the 1970's but as you will see later, the hypertext community still hasn't fully recognised that this is a problem that has been studied and solved before, albeit in a different context.

Another advantage of HyperCard was that it was quite extensible through its relatively easy to learn scripting language HyperTalk. For programmers this provided a way to make HyperCard do almost anything you wanted it to do, although at the time HyperTalk was described by one critic as "a COBOL programmers attempt to design object-oriented BASIC", which gives some indication of the type of programs that were written using it.

1.5 Hypermedia Issues in the 1990s

Although only available on Apple Macintosh machines, HyperCard set the standard that any commercial hypertext system had to beat as we moved into the 1990's, the decade of the CD-ROM, multimedia pc's and the Internet. For a number of years, the words HyperCard and hypertext became synonymous in the eyes of most users, which some argue put the development of real hypertext systems back many years and moved us further away from the visions of Bush, Engelbart and Nelson, rather than closer. Nonetheless, it brought the language of hypertext - buttons, links, nodes etc., - into everyday computer jargon and paved the way for what was to follow with the World Wide Web.

But the hypertext research community was moving on and as far as system building was concerned there were three different but by no means unrelated focuses. Firstly the standards community were looking for common methods of modelling and exchanging information between different hypertext systems, leading to the definition of models and languages such as Dexter [13] and HyTime [14] Another community was concentrating on developing tools to provide access to information available on the Internet- the global network of computers - such as the World Wide Web work at CERN [15] and Hyper-G at the University of Graz in Austria [16]. And a third community, of which the research group at Southampton were very much part, were developing so-called open hypermedia systems and link services, but more of that later.

First let us consider the phenomenon that became one of the major driving forces of the information revolution - the World Wide Web. Everybody reading this paper will have either used the World Wide Web or at the very least have read about it in the newspapers, more probably the former. It was developed by an Englishman, Tim Berners-Lee, working at CERN with a Belgian physicist, Robert Cailliau, from about 1989 onwards. They developed it in order to help the scientists at CERN organise the enormous amounts of information they generated and share that information with other groups around the world. It was really part of building and maintaining the organisational memory of the work at CERN, very much following in the footsteps of Bush and his vision of the Memex machine.

It remained a largely obscure research based system until it was popularised through the development of the Mosaic interface, which was released by the University of Illinois in 1993. This easy to use interface, whilst not realising the full potential of the system developed by Tim Berners-Lee, became almost overnight the easy to use hypertext interface to the information on the Internet. The developers of Mosaic left Illinois to establish Netscape Inc and the rest as they say is history.

The World Wide Web essentially created a universal hypertext system in three years, when Nelson hadn't succeeded with Xanadu in 30 years. The main reasons were that the Web and its browsers were free to use, it was pioneered by academics on the (essentially free) Internet, used open protocols so anyone could develop tools that used the system, provided a distributed cross-platform environment, was easy to use and the time was right!

Ted Nelson describes the Web as HyperCard on the Internet. He is essentially right in terms of the way it is commonly used. Just as with HyperCard, the highlighted words are buttons that can be clicked on by the user which activate a link that takes them to more information about that subject. The main difference being that the information could be anywhere on the Internet rather than just on your local machine. But there is a lot more to the Web than that and it has realised some of the dreams of the early visionaries. For example, people who publish lists of their favourite WWW sites could be described as very simplistic version of Bush's trailblazers, and today's Internet cafe's are in essence Nelson's high-street Xanadu kiosks.

The Web has shown us that global hypertext is possible, but it has also shown us that is easier to put rubbish on the net than anything of real and lasting value. In its current form it also encourages the development of unstructured, unmaintainable, unreusable and uncustomisable hypertexts. It is clear that authoring effort and the management of links are major issues in the development of large hypertexts. This has led to the design of systems which separate the link data from the document data thus enabling the information about links to be processed and maintained like any other data rather than being embedded in the document data. Research effort has also been concentrated on the development of link services that enable hypermedia functionality to be integrated into the general computing environment and allow linking from all tools on the desktop. The hypertext management system then becomes much more of a back-end process than a user interface technology. Such systems are usually referred to as open hypermedia systems although much use and abuse has been made of the term "open" in this context There are those who use the term to mean that the application runs on an open system such as UNIX: in this sense the World Wide Web is an open system. However, this is not what is meant in the context of open hypermedia.

There is not room to give a detailed description of what characterises such systems here but their capabilities will be discussed in later sections in this paper. The interested reader should look at Davis et al [17], Wiil and Osterbye [18] and Hall et al [19] for details. In the remainder of this section, we shall discuss the Microcosm open hypermedia system [17,19] as an example, including the motivations behind its development and its application to digital libraries.

1.6 Hypermedia and Digital Libraries: Microcosm and the Mountbatten Archive

One of the main motivations behind the development of the Microcosm open hypermedia system was the problem of providing hypertext support in large scale digital libraries. This was sparked when in 1987 the University of Southampton acquired the archive of the Earl Mountbatten of Burma from the Broadlands Trust. The archive contains about 250,000 text documents, 50,000 photographs, many of his speeches recorded on 78 rpm records, and a large collection of film and video. The archive spans his entire lifetime (1900-1978) and essentially mirrors British history in the twentieth century. It is also very multimedia in nature, composed as it is of much unstructured information in the form of free text, photographs, audio and video.

The multimedia nature of the archive makes publication and access impossible by traditional methods. For the forseeable future, there are no developments in technology that can reduce the enormous effort required to catalogue all the items in the archive, this requires specialist knowledge, but developments in information systems, both hardware and software can make it easier to store and organise the catalogues. Also, we can envisage a time when the information in such archives is available in digital form for anyone to make use of.

Creating a digital version of the archive is not something that fits neatly into a database. There is no linear sequence to the material other than that of chronological date, and users will want to move seamlessly from one document to other related documents. Every user is going to approach the material from a different perspective, so there is a need to create different 'views' for different users. from a hypermedia perspective this means the creation of different sets of hypertext links. Hence the need for an open hypermedia system/link service that stores links separately, allows those links to be applied across data of any media type and format across a network of heterogeneous platforms, and supports multiple users, allowing each user, or group of users, to maintain their own private view of the objects in the system.

We started in 1988 with the period when Mountbatten was in India, and we quickly learnt that the biggest issue was that of copyright. However, working with the archive helped us learn what was required of the next generation of hypermedia systems and indeed what was required of a multimedia information system in general. We needed to integrate hypermedia technology with database management and information retrieval systems to even begin to tackle the issues.

Structured access to the information through the use of document and database management systems is needed to answer questions like

“List the photographs taken by ....”

To support similarity matching across indexed documents to answer questions like

“Find photographs similar to ....”

we need information retrieval techniques, which is of course pretty hard when you are dealing with anything other than textual information. And you need hypermedia techniques to enable sophisticated browsing and cross-referencing capabilities to answer questions like

“Tell me about the people in that photograph ....”

The philosophy behind the design of the Microcosm model was aimed at creating a system that not only provided an open hypermedia link service, but that seamlessly integrated that service with database and information retrieval functionality, and in fact any other information processing capability that was required for a particular application area.

1.7 The Microcosm Model

The three-layered model of the Microcosm system [17, 19] has not really changed since its conception in 1989 although of course the way it is implemented has. The three layers essentially represent the way the user interacts with the system, the hypermedia functionality of the system, which is where the links are created and stored, and the information storage layer of the system. One of the key tenets of the design was that links could be defined on the basis of the content of an object in a document and/or its context, and not just on the basis of the position of a button within a document [20]

For example, suppose in working on the digital version of the Mountbatten archive we want to define a link that tells readers who Gandhi was. The word Gandhi appears all over the documents in the archive, so we don't want to have to manually define a link from the word Gandhi to say his biography or a picture of him, we want that link to automatically be available if the reader needs it. From this sort of idea the concept of the generic link evolved. This is a link that is available at any occurrence of a particular object, such as the text string Gandhi, wherever it appears. The scope of the link can be restricted as the author requires: for example we would probably put the generic link that associates the text string Gandhi with his biography into a link database designed for novice users, rather than one designed for expert researchers. A natural extension of this type of concept is that people will exchange and publish link databases in the same way that they exchange and publish linear texts at the moment.

The other main contribution that Microcosm has made to hypertext research, is the provision of a set of communication protocols that allows its links to be accessed through any viewer or application program that can communicate with the link service. This enables the integration of filters, or processes, that provide a framework for the dynamic generation of links through database, information retrieval or knowledge based techniques. In today's terminology such filters might be known as agents, an idea that we shall discuss in more detail later.

An example of a filter that dynamically generates "links" in the current version of Microcosm is the compute links filter (see [19] for details). The user is able to query the system to "compute links" based on a particular text selection, and the response is in essence a list of (pre-indexed documents) that contain that text selection. Because the filter architecture is highly configurable, this in-built text retrieval process can be replaced by any third party text retrieval process that can talk to Microcosm through its API. This facility, very much equivalent to the use of a search engine in a WWW context, is useful for authors as well as users to help them make links. But of course not all links can be made in this way, for example if we wanted to make a link from the text string Gandhi to a picture of Gandhi, that picture could not have been found by a standard text retrieval process. The author must either be using a more sophisticated information retrieval package or know where such a picture can be found.

Generally speaking the associative hypermedia links, i.e. links that associate content in the same or different documents, that we store in databases, or indeed that might be embedded in the documents in a different hypermedia system, represent some "knowledge" about the content of the documents on the part of an author. The destinations might have been system-computed, such as through the use of a text retrieval process, but the author, or indeed in some cases the system in the case of say knowledge based processes,, has decided that the association thus generated is worth storing in a database for future reference. The se links, whether manually or system generated, are stored in databases because the format for a link is a structured object, and the most efficient way to store and retrieve them is therefore through a structured database. This leads us very neatly into the following section where we discuss making links in structured data using relational database systems.

2 MAKING LINKS IN RELATIONAL DATABASE SYSTEMS

2.1 Relational Database Systems

Considerable quantities of data used to run businesses are held in relational databases. Examples of large databases include a 2.4 terabyte (a terabyte is one million megabytes) relational database at a retail store in the United States with 20 billion rows, supporting 650 concurrent users and 150 million transactions a day [21].

Much of the code in relational database management software is to provide reasonable and predictable performance, data integrity and for reliability, security and the ability to scale when more users access the systems, or when the volume of the data grows. Commercial relational database systems include DB2 [22], Informix [23], Oracle [24], SQL Server [25] and Sybase [26].

Relational databases were preceded by a number of other types of database managers, in the late 1960s and 1970s, that organised data in hierarchical and network structures. These databases included IMS from IBM and various CODASYL based databases. Navigating across links between parent and child data items was an essential part of accessing and updating data in these types of databases, some of which are still in use today.

Application programs had to be intimately aware of the underlying data structure. When the way the data was organised changed, the way the data was linked together also changed, and hence the programs had to change too.

The first paper to propose the relational model for managing data appeared in 1970 by Ted Codd [26] The relational model then evolved during the 1970s. The elements of the model include:

Data Structure: Organising data in tables, also known as relations, and no longer in networks or hierarchies. Each table has distinct rows whose order is immaterial.
Data Manipulation: Operations that can be performed by programs on the tables. The operations are set based and do not require a program to explicitly navigate across predefined links.
Data Integrity: Integrity constraints on the rows (the relational data) when the tables are updated.

In the relational model, all links between data items held in tables occur by comparing values. In the underlying relational database implementations, indexes may be built to speed up value comparisons.

Most of the early research for relational database systems was done in the 1970s and 1980s in projects such as System R in IBM [27]. The concepts of normal forms, (ways of structuring information to ease access and modification), were also devised in the 1970s [28]. The first relational database implementations in the late 1970s could not use more than one disk to store the data. Later multi-disk and distributed relational databases became usual.

Relational database management systems store information in a way that minimizes redundancy. Any duplication is strictly controlled to ensure data consistency throughout a database. The assumption is that users of the information held in a database will access it mostly with programs that have been specifically built for a purpose, although portions of code can be shared or reused. For example, you do not access your bank account information using the same application software that you use to query your electricity bill. There is an expectation that the programs will modify the data and that changes have to be made in a controlled manner. You would be unhappy if you lost some money when it was being transferred from one bank account to another because of a system or power failure, or because the receiving system was too busy to receive the money.

Relational databases usually offer programming interfaces to query and update the contents of databases including the database metadata (the table definitions) such as:

SQL: A precompiled application programming interface (the format of the parameters has to be known in advance) that can be embedded in many programming languages.
ODBC: A call (dynamic) application programming interface that can be used from many programming languages
JDBC: An interface for Java programs

In addition, relational databases offer system interfaces such as two phase commit participant interfaces for the X/OPEN XA transaction support [29] to ensure consistent updates across heterogeneous databases from a single application. Typically a database management system provides integrated interfaces for programs but not necessarily for human users.

2.2 The Data Definition Language

In general, information held in a relational database is about **entities** (or **things**) such as people, places, books, music, food, flowers, which have **attributes** (or **properties**). Each entity type is represented by pattern or template called a **table definition** and the language used is called **Data Definition Language** (DDL). Here is an example of a table definition, in other words the **metadata**, for an opera:

```
CREATE TABLE

OPERA

(OPERA               CHAR(50) NOT NULL PRIMARY KEY
                     /*opera name*/

COMP                 CHAR(50) NOT NULL PRIMARY KEY
                     /*composer last name*/

COMP_FN              CHAR(50)
                     /*composer first name*/

COMP_DOB             DATE
                     /*composer date of birth*/

COMP_POB             CHAR(50)
                     /*composer place of birth*/

1st_PERF             DATE
                     /*first performance date*/

1st_PERF_LOC         CHAR(50)
                     /*first performance location*/

OPERA_PHOTO          BLOB(10M) NOT LOGGED)
                     /*photo of opera*/
```

Usually a database is made up of a number of tables. A table that has been populated with data is made up of a number of rows. Here is an example of some rows in the opera table:

Some rows in the opera table

OPERA	COMP	COMP_FN	COMP_DOB	COMP_POB	1st_PERF	1st_PERF_LOC
Aida	Verdi	Giuseppe	10Oct1813	Parma Italy	24Dec1871	Cairo
Meistersinger	Wagner	Carl-Friedrich	21Jun1868	Leipzig Germany	21Jun1868	Munich
Rigoletto	Verdi	Giuseppe	10Oct1813	Parma Italy	11Mar1858	Venice
Turandot	Puccini	Giacomo	22Dec1858	Lucca Italy	25Apr1826	Milan

Each table, and thus each row, has a **primary key** which uniquely identifies each row. In the example above OPERA_TITLE and OPERA_COMPOSER together form the primary key in the opera table. The same opera title can be used by different composers. Indeed a single composer can produce the same opera in different forms, such as Don Carlos by Verdi, and so it might be appropriate to further qualify the primary key with a date. In commercial systems, it is usual to devise special identifiers (surrogates) for entities such as people to ensure uniqueness. The social security number is often used in the US to denote a person.

A table, and thus each row in it, can contain values for a data item that is a primary key in another table. In this case, the data item is called called a **foreign key**. The OPERA_COMPOSER could be a primary key in another table, such as a composer table. Foreign keys are one of the ways links are made between various tables in a database. A table can contain multimedia data items (Binary Large OBjects or **BLOB**s) as illustrated by the opera photo in the opera table above.

2.3 The Normal Forms for Relational Data

During the 1970s and early 1980s many papers and books appeared on the basic issues of modelling the real world in computer systems. Topics such as the fundamental difference between an entity, an attribute and a relationship (a link) were discussed at length in conferences, papers and books. Deciding how to arrange data items into tables became a research topic. Data and Reality by William Kent [31] begins with the statement: "**An information system, (e.g., database) is a model of a small, finite subset of the real world**". It also includes sections with titles such as:

Do records represent entities or relationships?

Are relationships entities? Are attributes?

In the 1990s, with the increased of availability of data in electronic form, in some ways the information system itself has become the real world.

When designing and developing an operational database application, a number of tables are implemented. To ensure that the tables do not become inconsistent when they are updated, it is advisable to follow the procedure of putting the entities and their corresponding attributes into third normal form before creating the tables. Here are some normal forms:

First normal form: There exists one value in each key or attribute position, and not a set of values.

Second normal form: Each attribute describes a fact relating to the whole primary key.

Third normal form: Each attribute is independent of the other attributes in the table.

There are a number of other normal forms, such as **Boyce Codd normal form** and **Fifth normal form**. You can find more details in [32].

The opera table above is not in third normal form, as the composer date and place of birth relate to part of the primary key (the composer name). Thus, the table violates second normal form. The opera table can be decomposed into an opera table and a composer table which are both in third normal form.

Some rows in the normalised composer table

COMP	COMP_FN	COMP_DOB	COMP_POB
Verdi	Giuseppe	10Oct1813	Parma Italy
Wagner	Carl-Friedrich	21Jun1868	Leipzig Germany
Puccini	Giacomo	22Dec1858	Lucca Italy

Some rows in the normalised opera table				
OPERA	COMP	COMP_FN	1st_PERF	1st_PERF_LOC
Aida	Verdi	Giuseppe	24Dec1871	Cairo
Meistersinger	Wagner	Carl-Friedrich	21Jun1868	Munich
Rigoletto	Verdi	Giuseppe	11Mar1858	Venice
Turandot	Puccini	Giacomo	25Apr1826	Milan

Integrity constraints can be imposed between the opera and composer tables through primary and foreign keys. For example, a constraint could be that it is impossible to delete a particular composer from the composer table while there are still operas composed by that person in the opera table.

2.4 Using Join Operations and Creating Views

Rows from different tables can be combined together using a **join** operation with SQL, ODBC or JDBC. Here is an example:

```
        exec sql select
        OPERA.OPERA, OPERA.COMP, OPERA.COMP_FN, OPERA.1ST_PERF,
OPERA.1ST_PERF_LOC, OPERA_PHOTO, COMPOSER.COMP_LN, COMPOSER.COMP_FN,
COMPOSER.COMP_DOB, COMPOSER.COMP_POB,
        where OPERA.COMP_LN = COMPOSER.COMP_LN and OPERA.COMP_FN =
COMPOSER.COMP_FN
```

This join request combines the two normalised tables OPERA and COMPOSER and returns the rows that formed the original OPERA table before normalisation.

A **view** is virtual table that is derived from one or more underlying tables. A view is defined using DDL and for example can be the result of joining two or more tables.

2.5 Using Transactions

Transactions are a collection of update requests, delimited by a **begin_transaction** and a **commit_transaction** statement, to one or more databases. Some database systems support **chained transactions** where an SQL request is always in a transaction, so when a commit request is issued, the next transaction starts immediately, and there is no need for a begin_transaction. Transactions have the following properties:

Atomicity:

All updates in a transaction either happen or don't happen. A transaction coordinator, which may be part of the database system or a separate piece of software, is used to ensure the atomicity of a transaction through the two phase commit (2PC) protocol.

Consistency:

The transaction leaves the database in a consistent state ready for other transactions to access the database. It is the application program that determines the precise point at which the data is consistent and not the database system, although it can help with ensuring the database remains consistent by checking integrity constraints

Integrity:

The transaction executes as if it is running alone with no other transactions. The database manager implements a locking scheme to ensure this behaviour.

Durability:

When the updates have been committed, they cannot be undone or rolled back. The database manager ensures the transaction is durable.

The properties are usually abbreviated to ACID and hence the term ACID transactions.

Transactions are very useful for maintaining consistency in databases, for example when bidirectional links are required through primary and foreign keys. Transactions ensure that partially updated links are not visible to others until the databases have become consistent. For example, when inserting an opera for a new composer, both the opera and composer information are added in two updates: insert an opera row and insert a composer row. To maintain consistency, both updates could be performed in a single transaction. A more usual and necessary example of transactions is when money is being transferred from one account to another: delete from one account and insert into the other account. An ACID transaction ensures that both updates happen together or neither of them happen, thus, the money does not get lost in transit between the accounts. For more information on transactions, please see [33].

2.6 Federated Databases

Federated databases make it possible to access diverse databases together that were not built with the intention of being integrated. Federated databases can span relational and non-relational databases.

An example of commercially available software that supports federated databases is DataJoiner [34] from IBM. DataJoiner provides a way, using regular SQL requests, to access relational data regardless of its location across various database managers such as DB2, IMS, Oracle, Sybase etc. Each of these database managers has its own way of describing its metadata. Each database manager has its own names for its databases which may coincide with names in other database managers. DataJoiner makes it possible to map the table names and views to nicknames which programs can use to access the tables across databases in a consistent way using SQL. The nicknames are stored in the DataJoiner catalog.

With DataJoiner, it is possible to perform join operations on data held in heterogeneous databases such as Oracle, Sybase and Informix. DataJoiner tries to optimise the performance of the join request by taking into account numbers of rows, use of indexes, processor speeds, in order to determine the fastest way to complete the operation. It is also possible to performed coordinated updates across various databases with DataJoiner using the two phase commit protocol for transactions. DataJoiner does not support cross database integrity constraints which have been described in research papers such as [35].

It is also possible to construct a form of federated databases using **middleware**. In other words using gateways and/or ODBC software to access a collection of databases. However, this approach would not provide integrated join operations with performance optimisations. See [36] for more information on ways of creating federated databases.

2.7 Other database activities

In addition to regular **relational databases** and **federated databases**, there are a number of other types of database activities including:

Object Oriented Databases

Object oriented databases first appeared in the 1980s as an extension to object oriented programming environments. Object oriented databases make it possible for users to define and manipulate their own data types, rather being restricted to tabular structures. In one respect, object oriented databases are similar to the original hierarchical databases, in that explicit hierarchical links are maintained. However there are many features in object oriented databases that are not available in other database types such as support for inheritance, long running transactions and version support.

In general, object oriented databases are not used in high performance operational systems but they are used in small scale systems such as design applications. See [37] for more information on object oriented databases.

Object Relational Databases

A hybrid object relational approach has become prevalent with the intention of overcoming the performance problems of object oriented database systems. Object relational systems are relational database systems enhanced with object modelling capabilities that provide the best of both relational and object databases. They support SQL with object extensions for accessing complex data types such as images, video and audio while attempting to provide the performance of relational systems.

Link support in object relational systems is the same as that for traditional relational database systems, and that is through comparing values of data items in the database rather than predfined hierarchical links.

Data Warehouses

A data warehouse is a collection of data used for corporate strategic decision making. The data is loaded from operational databases together with a record of the time it was collected. The data held in the warehouse itself is not usually updated. Instead updates to the operational databases are replicated in the data warehouse at various intervals. A data warehouse can be built for a whole organisation (top down). Subsequently, departmental information can be selected. Alternatively departmental data warehouses, known as **datamarts** can be created (bottom up), which can then be used to generate the corporate data warehouse. Very often data is de-normalised when it is held in a data warehouse in order to make it easier to discover previously unknown relationships between data attributes. **Data cleaning** operations are usually required before analyzing a data warehouse. Examples of data cleaning include removing duplicates caused by people spelling their names or addresses in inconsistent ways.

Data Mining

Various unsupervised learning techniques, known as data mining, can be applied to large collections of data such as data warehouses or extracts from relational databases to discover links and associations between data items or attributes. Previously unknown patterns can be discovered through grouping attributes into categories of varying granularity and checking for relationships between the categories. For example, for sales of ice cream, there could be an association between area (by country, state, city, street etc.) and time of purchase (morning, afternoon, evening etc) and age of customer (under 10, 20, 30 years etc.). **Data visualisation** methods can also be used, by grouping information into various categories and producing representation in various graphical formats, such as bar charts. Clusters can also be detected by plotting diagrams of the data items in the warehouse in various combinations.

OLAP (on line analytical processing)

OLAP tools are hypothesis driven in contrast with data mining tools which discover associations and patterns. OLAP is used when a question is asked, and multi-dimensional representations are used to check the results such as the number of opera CDs sold in a particular area per week to a particular age group. Please see [38] for an introduction to data mining, data warehouses and OLAP.

See [40] for information on future directions in database systems.

2.8 Ways of making links in relational databases

Links are made in databases in a number of ways:

Through metadata information when a database is being designed

For example, a foreign key provides a many to one link between rows in the table that contain the foreign key and the row that contains the corresponding primary key. It is possible to use a relational database to make links and maintain their consistency between files held outside the database, such as video clips and images [41].

Through the use of SQL, ODBC or JDBC

For example by performing a join across the rows of tables that have been populated. The join operations rely on metadata and matching the values of the attributes being joined. Indexes may be used to speed up the performance of join requests. Although there is a strict separation of metadata and data, it is possible to perform SQL operations on the metadata itself.

Through link discovery after a database has been populated

For example data mining can be used to detect clusters in certain combinations and categories of data items. The clusters can be represented graphically and it is possible to **drill down** within the clusters or categories to detect links at a more detailed level.

3 MAKING LINKS IN THE WORLD WIDE WEB

3.1 The World Wide Web

The World Wide Web consists of the collection of Web servers on the Internet together with clients, often Web browsers, that access them. Web servers support the HTTP (Hypertext Transfer Protocol) communication protocol which is used to deliver HTML (Hypertext Markup Language) pages to the client. These pages can be created in advance, known as **static pages** or at the time the client requests a page along with a set of parameters identifying the type of page needed, known as **dynamic pages**. The World Wide Web was first developed at CERN. See [42] for more information.

3.2 The Universal Resource Locator

A flexible universal naming scheme (Universal Resource Locator or URL) is used on the Web which means that any piece of information held on a server can be located from any client on a TCP/IP network: the Internet or an **Intranet**. An Intranet is a TCP/IP network that is specific to an organisation and that is usually separated from the Internet by a **firewall computer**. The firewall monitors all inbound and outbound network requests and permits only the requests that conform to policies defined by the firewall administrator.

Here are two examples of URLs:

Example 1: *http://www.w3.org/pub/WWW/People/Berners-Lee*

This is the URL requesting Tim Berners-Lee's home page at the w3 consortium. HTTP is the protocol used and www.w3.org is the server name running the Web server. The Web server subdirectory that contains the HTML with information about Berners-Lee is pub/WWW/People/Berners-Lee/ .It is likely that the URL above refers to a page that was created before the request was made, i.e., to a static page.

Example 2: *http://www.altavista.digital.com/cgi-bin/query?pg=q&what=web&fmt=.&q=%22BernersLee%22*

This is a URL requesting that the AltaVista search engine checks its indexes and returns a page that contains a list of all the pages that contain “Berners-Lee” somewhere in the text. For the request above to be sent to the server, the AltaVista user would have typed in the text "Berners-Lee" into an HTML form displayed on the browser. The input fields are delivered to the AltaVista server which would then run a program passing it the user input value "Berners-Lee". The program would create and deliver the page containing the list of "Berners-Lee" pages to the AltaVista server which would return it to the browser.

When a browser sends an HTTP request to a server to retrieve a page, the server name portion of the URL is mapped onto a 32 bit IP address. For example, www.w3.org corresponds to : 18.23.0.22. With the growth of the Internet, there is concern that the 32 bit IP addresses will run out early the next century. One of the objectives of IPv6 proposal [43] is to deal with this problem by introducing a more sophisticated addressing scheme that requires 128 bit addresses.

The universal naming scheme is one of the essential ingredients for the success of the Web, and has been extended to incorporate many protocols and interfaces such as JDBC (the Java DataBase interface). The format of a JDBC URL is:

jdbc:<subprotocol>://hostname:port/database-name such as

jdbc:oracle://www.opera.com/verdi-database

See [44] for more information on URLs.

3.3 Hyperlinks in HTML

HTML hyperlinks, together with search engines which are described later, make the content of the World Wide Web appear integrated even though Web servers are created and managed autonomously. Here is an example of how a hyperlink can be included in an HTML page:

> *<A href="http://192.107.38.102/java/jag/jag.html">James Gosling</A> attended the opera yesterday*.

The URL is not displayed and the text appears as follows:

> *James Gosling attended the opera yesterday.*

The user can click on the text "James Gosling" (referred to as an anchor or hotspot) to ask the browser to retrieve the page with the corresponding URL. No predefinition nor metadata is required in any database to describe the link, e.g., as a foreign key, and the creator (human or program) of a Web page can include hyperlinks anywhere in a page. It is not necessary to create an index on the hyperlink anchor or on the URL in order for a person or program to navigate between pages using hyperlinks.

It is also possible to create hyperlinks that refer to dynamically created HTML pages. For example:

> *Here is <a href=*
> *"http://www.altavista.digital.com/cgi-*
> *bin/query?pg=q&what=web&fmt=.&q=%22Berners-Lee%22"&>*
> ***a list**</a> of Tim Berners-Lee pages.*

In the case above, the user would click on the text "*a list*" to retrieve a dynamically created list of pages.

It is possible for the anchor to be an image. For example:

```
<a href="http://www.w3.org/pub/WWW/People/Berners-Lee/">
<img src="http://www.w3.org/pub/WWW/People/Berners-Lee/tim.gif</a>
attended the ballet yesterday.
```

Web servers typically do not concern themselves with the consistency of the data to which they provide access. Nor, do they concern themselves with the client system other that it communicates through HTTP. It is unlikely that the Web would have been successful if it has imposed the type of integrity checks that are common in relational databases. For example, it is usually possible to delete an item on the Internet while there are still hyperlinks which point to it. Thus, hyperlinks between Websites can be particularly volatile. However, there are types of software, known as link checkers, which check the validity of all the outbound hyperlinks from a particular page or Website. (A Website is the collection of HTML pages and programs relating to a particular Web server.)

3.4 Java and the Web

Java is a general purpose object oriented language [45]. Sun Microsystems began the development of the Java language and the Java Virtual Machine (JVM), initially known as Oak, in 1991 to provide an environment that would work well with electronic household devices. Thus, the language had to be compact and tolerant of frequent changes in underlying technology. Instead of compiling Java into the target system machine code, Java is compiled into a concise intermediate language known as Java byte codes. Java programs are then distributed in Java byte code form and processed by the JVM at runtime on the target system. The JVM interprets and executes the Java byte codes and acts as the interface between the Java program and the target operating environment. A compiled Java program can run on any system, provided that the JVM exists on that system.

A major breakthrough in the Java project occurred in late 1994 when Sun built the HotJava browser. Sun incorporated the JVM within the HotJava browser software, thus enabling HotJava to run compiled Java programs (that is, in byte code form) known as applets. Sun introduced a new HTML tag, <APPLET> so that HotJava could retrieve applets from Web servers by using the HTTP protocol across the Internet. Here is an example:

```
<APPLET CODE ="HelloWorldApplet.Class" height=100 width=200>
We are sorry but you are not running a Java capable browser
</APPLET>
```

The idea of downloading Java applets spread quickly and the JVM was incorporated into a number of browsers and operating systems. Java is widely used for programming applets and regular applications that are not invoked from browsers. Java has had a major impact on the Web from the user interface perspective. Java applets can be run in the Web browser to perform a variety of activities including communicating directly with server systems to retrieve data from relational databases. Java is also used to write server systems such as the Jigsaw Web server [46].

Note that the portion of the Web page that displays the Java applet output is separate from the section of the page that displays HTML. Indeed, the Java applet does not have access to contents of the HTML page within which it is included, other than the parameters that are explicitly passed into the applet. The Java applet does not have access to any HTML forms input.

3.5 Accessing relational databases from the Web

The widespread use of Web clients (often Web browsers) caused many software developers to build Web based software that provide access to databases [47]. These are the most usual ways to access relational databases from the Web:

Server Side Programming

Many Web servers offer a programming interface that enables gateway code to be invoked when a user request for a Web page arrives at the server using HTTP. The request usually includes parameters that the user typed into a Web form. The gateway code can then access a database in accordance with the input parameters and create an output HTML page which it delivers to the Web server which is then delivered to the browser using HTTP. One of the first interfaces of this type supported by Web servers is called the Common Gateway Interface (CGI) and gateway software to provide Web access for relational databases started appearing in 1994. The CGI turned out to be rather inefficient on a busy server as it initiates and terminates an operating system process on every user request, although the CGI is very well accepted and used on many system. Other interfaces for Web servers have been devised such as the NSAPI (the Netscape API).

Client Side Programming

With the availability of the Java Virtual Machine at first in Web browsers in 1995, and later in regular operating systems, it became possible to download Java client software to access a database directly or through an intermediate gateway from a browser. Thus, the Web server is the source of the client software, and HTTP is the distribution mechanism for the Web software. The Java client software can use its own protocol to communicate with its server, but there are advantages in using a standard Internet protocol, in order to simplify outbound navigation through firewall computers. Alternatively, a Java client can use JDBC to access and update relational databases. The Java client and gateway approach is similar to traditional client server programming, and related design issues apply with one major exception: The Java download mechanism ensures the most recent level of client software is used. At present, the latest versions of the appropriate Java classes are downloaded the first time the relevant page is accessed from an instance of a browser. Currently there is a browser restriction that a Java client can communicate only with the system from which it was downloaded. The reason for the restriction is to ensure that a Java client that has been downloaded from outside the firewall does not communicate with a server within the firewall, and hence is not able to transmit sensitive internal information outside the firewall.

Embedded HTTP

With this approach, the target system implements HTTP itself and maps it maps the protocol to its own interfaces. Although relational databases have not adopted this approach widely yet, a number of systems have done so. Here are some examples of systems that support HTTP:

Systems that use embeddable HTTP server software such as MicroServer from SpyGlass, ProWeb from 3Soft, and Jigsaw from W3C.

Devices that incorporate embedded operating systems such as Embedded Internet for pSOSystem from Integrated Systems.

Computer devices such as some Tektronix printers and Cisco Routers.

Groupware such as Lotus Domino and broadcast systems such as PointCast.

Transaction processing and database systems such as CICS.

3.6 Caching

The primary purpose of caching on the Web is to avoid transmitting data unnecessarily across the network. The server can indicate the items to be cached by intermediate servers that reside between between the client and the Web server. Caching can occur at many points and they include:

At the client: The Web browser cache

Near a group of clients or group of servers: The Web proxy cache

At the server: This can be similar to database caching where the main reason for caching is to avoid disk accesses. Dynamically generated pages can also be cached to reduce repeated generation.

It is possible for the browser to automatically refresh pages at regular intervals, e.g., every minute, if the HTML page includes the appropriate HTML tags:

<META HTTP-EQUIV="Refresh" CONTENT=60>

For more information on caching, please see [48].

3.7 Web Search Tools

With the large volume of data on the Web, the popularity of search tools has increased for the Internet and the Intranet. These are some categories of search tools:

Internet search tools: These tools include crawlers or agents (which can be human such as at Yahoo) or programs (such as the AltaVista or HotBot crawlers) that retrieve Web pages from the Internet. Local indexes are built and made available for users. The nature of the data indexed varies, for example Webcrawler indexes the significant sections of Web documents such as headings, whereas AltaVista indexes every word in a Web document. The presentation of search results is evolving and is a topic for research in itself. Some search tools will indicate only one of the pages on a Website that include the topic or keywords being sought but with an option to request further relevant pages from that site. Options are sometimes included to enable further similar pages to be retrieved but that may not include the exact words originally request. It is possible to search the Internet for images, and to isolate searches to particular countries or domains. Some agents, such as the staff at Yahoo, order and structure links between items on the Web in a way that could be considered to be in the spirit of Vannevar Bush's "trail blazers".

Intranet search tools: These tools often bypass the use of a crawler and the relevant indexes can be built on or near the Website itself, thus reducing the load on the internal network. Some Web servers are packaged with their own indexers.

Internet search for people: Some search tools provide an option to find people's e-mail adresses. The excite search engine has such an option.

See [49], [50], [51], [52] and [53].

There have also been a number of projects to assist with measuring and visualising the Web. For example, in 1996 Tim Bray [54] illustrated Websites as towers. The height of each tower represents the number of inbound links to the Website, thus a tall Website is very visible. The size of the ball at the top of the tower represents the number of outbound links, thus a Website with a large ball is very luminous. There are difficulties with this representation when HTML pages are created dynamically and the link information is held in indexes and databases whose structure is known only to the software running at the site itself. Yahoo appeared to be a very luminous site as at the time it had static HTML pages containing links. AltaVista did not appear to illuminate the Web as most of its pages are created dynamically in response to specific user queries.

3.8 Making links in Web based relational database applications

There are a number of differences between setting up a Website and a relational database system that arise from the differences in levels of integrity supplied by the underlying software. Integrity constraint and transaction support costs in both human effort to define and set up the relevant metadata, and in computer instructions executed. Excluding communications, the number of computer instructions to retrieve a Web page from a Web server is between 1,000-10,000 and the number of computer instructions to retrieve a row from a relational table is between 10,000-100,000. The additional instructions are the cost of data consistency, reliability and integrity. For serious applications on the Web, using a relational database to store business data and links between data items, including files outside the relational database itself can provide the reliability that would not be possible from a Web server alone.

In HTML, links are made between Web pages through URLs at HTML page development time for static HTML, and at HTML page generation time for dynamic HTML. Simple tools such as regular word processors can be used to generate links between Web pages without professional programmers being involved. In contrast, to make links in databases it is usually necessary to involve professional programmers and database administrators. The links in HTML are similar to the original navigational databases but without integrity. However, navigation using HTML is intended for humans rather than programs.

Locating data in a relational database usually requires site specific knowledge. Names can overlap with each other across sites possibly causing ambiguities if an attempt was made to integrate a collection of existing relational databases. When locating an item on the Web, the name of a Web data item (the URL) coincides with its address. By knowing the name, the item can be found relatively quickly through the DNS (domain name service). In a relational database, it is necessary to know a number of pieces of information before being able to locate an item on the database.

The results of searches using the Internet search engines can be viewed as providing links between similar Web pages. Indeed, some search engines provide an option to retrieve pages that are similar to pages that have been retrieved already. There are also projects to enable users to add their own links to pages that they do not own, e.g. aqui [55].

By combining static HTML pages and dynamically generated HTML from the content of relational databases, for example through server side programming, it is possible to generate a variety of HTML links that represent relationships between:

Database primary and foreign keys [56]

Database keys or attributes and static Web pages

Database keys or attributes and dynamic Web pages

Portions of database Web programs, such as when navigating through a Net.Data [57] Web based database application that requires a series of user interactions

Categories and groupings of relational data, such as when drilling down in a Web based data mining or OLAP application

As relational database applications on the Web do not include static HTML pages, their content is not indexed by the regular Web crawlers. See [58] and [59] for projects to index and search heterogeneous data sources on the Web. There are also projects which attempt to integrate semi-structured data through mediators and through wrappers such as [60] and [61]. In the next section, more sophisticated methods for managing links in the future are described.

4 MAKING LINKS - THE FUTURE

4.1 Link Services and Digital Libraries for the Web

The growth of the Web has been phenomenal and it could be argued to be the killer technology of the information revolution let alone of hypertext. Suddenly everyone - journalists, industrialists, politicians, school children - want to have their own home page on the net. Most people who use computers at work or at home are now aware of the concept of hypertext as it appears in the Web. Due largely to the influence of the design of the early Web browsers, such as Mosaic and Netscape, common usage embeds Web links in the documents using HTML, and only allows the presentation of one document at a time. Link management is generally non-existent and spaghetti hypertext authoring is rife. Many individuals and organisations do not realise until it is too late how impossible it is to maintain and up-date information created in such an independent and autonomous fashion. This comes as no surprise to software engineers!

However, there is nothing inherent in the design features of the Web that says this has to be the case. It is actually possible to implement almost any hypertext model in the Web environment. Links don’t have to be embedded, Web document and link management tools can and are being implemented, and new browsers could allow multiple documents to be presented at one time, more sophisticated navigation features and integrated editing facilities. Indeed the use of frames in Web browsers has already led to the development of Web sites that are much more user friendly in their interface design. See the book by West & Norris for a discussion of such techniques [62].

As we have seen above, there has been much progress in the creation and management of large scale Web sites through the use of relational databases to store the pages and tables that represent the relationships between them, so that for example the intelligence concerning which page should be loaded next can be built into the database. However in the case of loosely structured digital libraries, there is often no concept of the next or previous page. To enable users to make full use of such information sources we need to provide the rich cross-referencing and linking capabilities described in Section 2 as well as the integration of other information processing tools such as text retrieval systems. This means we need to be able to support the large-scale creation and maintenance of associative links between the content of documents, as well as any navigational or structural links that are inherent in the design of the information system. Just as we separate the structural links from the data using a relational database, we need to store the associative (sometimes called unstructured) links separately from the data as well, as we described in Section 2. As a result a number of groups in the hypermedia community are working on the development of link services for the Web to support applications such as digital libraries [63, 64, 65, 66, 67].

One of the most well-known of these is the Hyper-G system, now available commercially as HyperWave [63]. Hyper-G is described by its creators as a second generation Web system because it represents an advance over the standard Web and supports tools for structuring, maintaining and serving multimedia data. In addition it guarantees automatic link consistency and supports links among multimedia documents, full-text retrieval, a UNIX-like security system, and client gateways to Gopher and Web browsers such as Netscape, Mosaic and MacWeb.

Like the Web, Hyper-G is based on a client server architecture but unlike most Web environments, it stores its link separately from the documents in an object-oriented database. In this way it can provide the document and link management tools to ensure consistency and integrity of links, as well as allowing different sets of links to be applied to the same set of documents. If the Web in its current form is like HyperCard on the Internet, then Hyper-G could be described as Intermedia on the Internet since although it supports standard Web browsers, its full functionality can only be accessed through dedicated Hyper-G viewers - the main current ones being Harmony for the X-Windows environment and Amadeus for Microsoft Windows. Also as in Intermedia, links are generally point to point. In text documents they can be created, as is common in HTML, by using an HREF reference to a URL. Using the Hyper-G authoring tools, Harmony and Amadeus, they can be created interactively by point-and-click methods with source and destination anchors in multimedia documents such as videos, PostScript and VRML documents. One of the main strengths of Hyper-G are the tools it provides for visualising the structure of the documents and the links in Hyper-G applications through graphical browsers. Like Microcosm, it also contains in-built search facilities, both structured (database query access to documents) and unstructured (full-text retrieval of indexed documents).

4.2 Microcosm and the Web: the Distributed Link Service

The original aim of the Microcosm project was to take the fully 'open' system approach and to provide a light-weight link service that is available to the user whatever information they are dealing with, whether it is information under their own or their organisations control, or external information over which they have no control, or indeed legacy data within their own environment. The Microcosm architecture also provides methods for alleviating the link management problem by enabling the development of links that do not simply link a point in one document to a point in another document, but can be dynamically created by whatever algorithm is most suitable for the information environment.

The Distributed Link Service (DLS) is a development of the Microcosm philosophy applied to the distributed environment of the World-Wide Web that has been developed in Southampton [67, 68]. In the same way that a client connects to a remote Web server to access a document, the DLS allows the client to connect to a link server to request a set of links to apply to the data in a document. There are several different link database categories supported by the system, at the most general level are server databases, which apply whenever the system is queried. Link databases may also be provided for a group of documents, or a particular document. In addition, a variety of 'context' link databases are available which the user may select from. By choosing a different context, the user may adjust the available link set to best suit their current information requirements. Like Microcosm, the system supports the use of generic links, which allows links to be applicable beyond the scope in which they were originally created and considerably reduces link authoring effort.

The link server facilities of the DLS were implemented first as CGI scripts invoked by a standard WWW server. The user could make a selection in a Web document, and choose an option such as Follow Link from a menu just as in Microcosm. However a major problem with this interactive client approach is the engineering requirements of producing and maintaining software that applies the available link services to a range of different viewing applications using a variety of WWW browsers on a range of different host operating systems. Hence an alternative, 'interfaceless' approach was investigated: to make the link service transparent to its users by embedding it in the Web's document transport system, compiling links into documents as they are delivered to the user by a specially adapted WWW proxy server.

This approach requires no extra client software for the user, which is an immediate practical benefit, but it does suffer from a number of disadvantages. Firstly, the loss of interaction makes it impossible to create a link by the usual method (in Microcosm) of making a selection and choosing Start Link from the menu. It also changes (perhaps for the worse) the browsing paradigm from 'reader-directed enquiry' to 'click on a predefined choice' [20]. Secondly, this behind-the-scenes link compilation is applicable only to documents which are delivered via the WWW and which are coded in well-understood document formats that can themselves support some form of hypertext link. These requirements abandon some of the advantages of the open system previously described, since there are relatively few document formats which can have links embedded. However, because of its engineering advantages, it is this version of the DLS that is currently being commercialised as Webcosm. A link control panel provides the user with configuration possibilities to overcome some of the disadvantages described above and interfaces that provide the full interactive capabilities of Microcosm are being investigated.

The DLS is being used in a digital libraries project concerned with electronic journals funded by the Electronic Libraries Programme (eLIB) of the UK Higher Education Councils' Joint Information Systems Committee (JISC). The aim of this project is to produce an open journal framework (OJF) which integrates collections of electronic journals with networked information resources [69].

The seamless integration of journals that are available electronically over the network with other journals and information resources is not possible with traditional paper publishing but has long been sought by the academic community. The emergence of the World-Wide Web and its freely available user, or 'browser', interfaces has dramatically simplified access to such resources and has raised awareness and expectations of electronic distribution of information and data, and has particular significance for journal development. In its present form, however, the Web is limited in its capabilities for page presentation, hypertext linking, user authentication and charging, but it defines a flexible framework into which more advanced technologies can be slotted. It is in these areas that the Open Journal Framework project intends to provide mechanisms that realize the wider potential for networked journals that has been created by the Web.

The philosophy of the project, at the most basic level, is to provide immediate access to electronic versions of existing quality journals, the information content of which includes scientific formulae, tables, diagrams and high-resolution colour photographs. Beyond that the aim is to provide powerful hypermedia linking techniques to allow naive users direct access to secondary information resources, instead of requiring them to use these resources independently. The links are created dynamically at the users' request and do not need to be explicitly embedded in the journal papers when they are authored, thus realizing the concept of the 'open' journal.

This is achieved by making use of current hypermedia technologies, for example Adobe Acrobat for presentation, and Microcosm, in the guise of the DLS, for information linking, as well as the World Wide Web. In addition, through the development of subject-expert agents implemented within the DLS framework, the user will be offered a greater range of resources than he or she alone would normally be aware of.

4.3 Making links in non-text data: content-based retrieval and navigation

The aim of the MAVIS (Multimedia Architecture for Video, Image and Sound) project is to extend the Microcosm architecture to facilitate the use of generic and dynamic links from all media types, not just text [70,71]. This is not just simply a question of making the selection content available at the authoring and the link following stages because the problem of matching is substantially more complex than for text.

For example, we would like to be able to author a generic link from an object in an image to another point in another document, perhaps some text about the object. If it is a versatile generic link, we should be able to follow the link from any instance of the object in any image in any document. This is not usually an easy task of course because the object may appear very differently in different images. We need to extract media based representations of the selection that are as invariant as possible. Ultimately we need to recognise the object using prior knowledge of how it appears in the image. We may then recognize it anywhere in other images to enable us to follow the generic link effectively.

With the current state of image understanding, such an approach would have to be application specific. However, there are some general features which can form the basis for useful generic link following mechanisms for images, video and sound. For example, we can use colour distribution, texture and out-line shape or combinations of these, for generic links in images and video, and the pattern of the sound wave for audio. The same techniques can be used to implement a *compute links* (the Microcosm text retrieval links described in Section 2) facility for non-text media, thus providing a method of content-based retrieval for images, video and audio which is seamlessly integrated with the hypermedia model. The extension of the Microcosm generic link facility to non-text media has the same potential to reduce the authoring effort as with text, and as well as providing a method of content-based navigation across all media types.

The matching between the query selection and the link anchor must be fuzzy since exact matching is clearly impossible in non-text applications. Measures of similarity are required to assess the quality of a match and to decide if a link should be followed. It has been argued that even for text, lexical similarity matching has problems because of the difficulty in capturing the semantics. This is even more of a problem for non-text media, but we believe that general purpose tools for content-based retrieval and navigation will prove to be extremely powerful, particularly when supported by filters that provide content and context based semantic analysis.

The MAVIS architecture is based on the development of a systems of signatures. In Microcosm a link allows the user to relate in a declarative way a selection in one document to a selection in another. MAVIS extends this concept by introducing the idea of a signature. A signature is a representation of the content of a selection, used in both link generation and following. In the case of a text anchor for a generic link, only one form of signature was required - a normalised representation of the original text anchor content. MAVIS, by contrast, allows many forms of signature to co-exist. In the case of images for example, the signature might represent the colour of the selection, or its shape, or its texture, or even a combination of any of these. The MAVIS architecture allows for multiple processed representations (signatures) of the original anchor data: each type of signature being handled by a separate signature module.

Multimedia information systems are one of the fastest growing areas of computer science, but substantial improvements in navigational tools for non-text information are essential if society is to reap the full benefit of the growth in multimedia information. One of the major challenges is to be able to navigate elegantly through non-textual information without having to manually associate textual keys. For image and video, versatile content based retrieval and navigation requires solutions to many of the problems of computer vision.

The next step is to develop more versatile and intelligent facilities for supporting content based retrieval and navigation for both text and non-text media building on the strengths of the MAVIS architecture. This will include the development of a multimedia thesaurus, which will not only accumulate textual representations of objects but also appropriate representations from other media, and the development of intelligent filters to work with the multimedia thesaurus, designed to improve both the indexing of media-based information during entry to the multimedia information system and classification and matching during link following. This leads us onto the idea of filters being developed into classes of intelligent agents as discussed in the next section.

4.4 Making links intelligently: filters become agents

During the initial design for Microcosm, it was expected that the combination of the generic link facility and other dynamic link generation processes would create a significant problem of information overload for Microcosm users. Hence the link processors were called *filters* because it was expected that their main task would be to reduce the number of links offered to users according to some appropriate algorithm. In fact, most of the intervening years have been spent designing and building filters to create links, such as the 'Compute Link' filter, so that the term became rather a misnomer for current usage of the system. However, things are now coming round full circle because the system has been developed to such a point that it is possible in practice rather than just in principle to use it to build large-scale applications with very large numbers of links, and processes are needed that truly act as filters to decrease the information overload for users.

In today's terminology, filters would probably be called *agents* because Microcosm can essentially be considered as a group of processes working together to perform tasks on behalf of the user, in other words a community of software agents.

The current Microcosm communication model allows isolated agent processes to be constructed and both Prolog and neural networks have been used to construct experimental agents within the current design. However the limitations in the design of the current Windows version of Microcosm mean that no direct communication between processes is possible. Experiments with a direct model for communication between processes have shown that this model supports co-operative agent processes. The extension of this model to construct a heterogeneous fully distributed version of Microcosm as part of the Next Generation (TNG) project, opens the way for extensive research into the implementation and use of co-operative agents within the new architecture fully integrated with other information systems available via the Internet [72]. A large subset of these agents will be doing fairly straightforward information processing tasks such as library management. These agents need not be intelligent and we tend to refer to them as "distributed information management" or DIM agents. However, the aim is to enable the user to customise the environment to suit their own purposes and to "train" the supervisor agents to this end. It is here that the intelligence of the system will lie.

The results of the TNG project are being utilised in an EU funded project, MEMOIR [73], which is aiming to build a corporate information system, where documents, links and user trails are stored in an object-oriented database, accessible to users via standard Web browsers and a distributed hypermedia system brokered by agent services. As users access the information in the system, and indeed external to it, they can keep a record of the trails of information they have followed. The MEMOIR system helps to match people with trails, trails with trails and people with people to try and overcome the problems of finding information in large corporations.

4.5 A Vision for the Future

We have all seem the science fiction films and TV series where the crew of the spaceship find the information they need through communication with an "intelligent" agent. The most famous example is probably HAL in the film *2001: A Space Odyssey*. It is possible to see now the sort of building blocks that are required to create an information processing environment such as HAL would need access to. Just as we “think” by making associative links between disparate pieces of information, so will the intelligent information processing agents of the future. They will draw on information from vast banks of distributed databases and document management systems. They will access this information by both structured and unstructured means using combinations of database, information retrieval and hypermedia techniques. In effect, all these technologies are about making links.

If we ask the question "Tell me about Gandhi" of an intelligent agent such as HAL, then first the agent must have some understanding of the context in which we are asking the question, in order to answer it with some degree of confidence. The next step maybe a search of a database of significant figures in history. If none is available then a general search of relevant documents may be made or brokered using search engine techniques. Or alternatively, someone may have asked this question before and the agent will "remember" the link and produce the information that successfully answered the question previously, or that is flagged as being a suitable answer to the question in an appropriate link database. The possibilities are endless, and of course all of them require the agent to intelligently prioritise and process the available information sources. At the moment the worlds of structured and unstructured data seem relatively unconnected but we hope we have shown here how they are increasingly coming together.

5 REFERENCES

1. The Hartlib Papers Project, http://www2.shef.ac.uk/hartlib/project.html.

2. Bush, V., As We May Think, Atlantic Monthly, pp 101-108, July 1945

3. Engelbart, D.C., A Conceptual Framework for the Augmentation of Man's Intellect. In Vistas of Information Handling, Vol. 1, Spartan Books, London, 1963.

4. Nielsen, Multimedia and Hypermedia: the Internet and Beyond, Academic Press, 1995

5. Nelson, T.H., Literary Machines. Published by the author, 1981.

6. Yankelovich, N., Meyrowitz, N. & van Dam, A., Reading and Writing the Electronic Book. IEE Computer 18 (10) pp15 - 30, 1985.

7. Akscyn, R., McCracken, D.L. & Yoder, E., KMS: A Distributed Hypertext for Sharing Knowledge in Organizations. Communications of the ACM, Vol 31(7), pp 820 - 835, July 1988.

8. Shneiderman, B., User Interface Design for the Hyperties Electronic Encyclopedia. In the Proceedings of the ACM Hypertext '87 Workshop. The University of North Carolina at Chapel Hill, pp 189 - 194, November 1987.

9. Halasz, F.G., Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. Communications of the ACM, Vol 31(7), pp 836 - 855, JUly 1988.

10. Yankelovich, N., Haan, B., Meyrowitz, N. & Drucker, S., Intermedia: the Concept and the Construction of a Seamless Information Environment. IEE Computer, Vol 21(1), pp 81 - 96, July 1988.

11. Brown, P.J., Turning Ideas into Products: the Guide System. In the Proceedings of the ACM Hypertext '87 Workshop. The University of North Carolina at Chapel Hill, pp 33 - 40, November 1987.

12. Goodman, D., The Complete HyperCard Handbook. Bantam Books, New York, 1987.

13. Halasz, F. & Schwartz, M. "The Dexter Hypertext Reference Model". Communications of the ACM, Vol. 37, No. 2, pp. 30 - 39, (Feb 1994).

14. DeRose, S.J. & Durand, D.G., Making Hypermedia Work: A User's Guide to HyTime. Kluwer Academic Press, 1994.

15. Berners-Lee, T.J., Cailliau, R. & Groff, J-F., The World Wide Web. Computer Networks and ISDN Systems, Vol. 24(4-5), pp 454 - 459, 1992.

16. Andrews, K., Kappe, F. & Maurer, H. "Hyper-G: Towards the Next Generation of Network Information Technology. Journal of Universal Computer Science, April 1995.

17. Davis, H.C., Hall, W,. Heath, I., Hill, G.J. and Wilkins, R.J. "Towards an Integrated Information Enivornment with Open Hypermedia Systems". In the Proceedings of the ACM Conferenc e on Hyerptext, ECHT'92, pp. 181 - 190. ACM Press, 1992.

18. Wiil, U.K. & Osterbye, K (eds.) "The Proceedings of the ECHT'94 Workshop on Open Hypermedia Systems, Edinburgh, 1994". Technical Report R-94-2038, Aalborg University, October 1994.

19. Hall, W., Davis, H.C. & Hutchings, G.A. "Rethinking Hypermedia: the Microcosm Approach". Kluwer Academic Press, Boston, 1996.
20. Hall, W. "Ending the Tyranny of the Button". IEEE Multimedia, Vol.1, No.1. pp. 60 - 68, 1994

21. http://www.dbpd.com/, Database Programming and Design, Volume 10, Number 6, June 1997

22. http://www.software.ibm.com/data/db2/, DB2

23. http://www.informix.com/, Informix

24. http://www.oracle.com/, Oracle

25. http://www.microsoft.com, SQL Server

26. http://www.sybase.com/, Sybase

27. A Relational Model for Large Shared Data Banks, E.F. Codd, Communications of the ACM, 13, 6, (June 1970), 377-387

28.System R: Relational Approach to Database Management, ACM Transactions on Database Systems, Volume 1, Number 2 (June 1976)

29. A Sophisticate's Introduction to Database Normalization Theory, C. Beeri, P.A. Bernstein and N. Goodman, Proceedings 4th International Conference on Very Large Databases (1978)

30. http://www.xopen.org, X-Open

31. Data and Reality, William Kent, North Holland, 1978

32. An Introduction to Database Systems, Chris Date, Addison Wesley

33. Principles of Transaction Processing, Philip Bernstein and Eric Newcomer, Morgan Kaufmann, 1996, ISBN: 1-55860-415-4

34. http://www.software.ibm.com/data/datajoiner/ Data Joiner from IBM

35. The role of integrity constraints in database interoperation, Mark W.W. Vermeer,
Peter M.G. Apers, University of Twente, Proceedings of the Very Large Database Conference 1996

36. Federated Database Systems for Managing Distributed Heterogeneous, and
Automous Databases, Amit Sheth and James Larson, ACM Computing Surveys, Volume 22, Number 3, September 1990.

37. Introduction to Object Oriented Databases, Won Kim, The MIT Press, 1990

38. Database Principles, Programming Performance, Patrick O'Neill, Morgan Kaufmann, 1994, ISBN:1-55860-219-4

39. Data Mining, Pieter Adriaans, Dolf Zatinge, Addison Wesley, 1996, ISBN: 0-201-40380-3

40. Strategic Directions in Database Systems-Breaking out of the Box, Avi Silberschatz and Stan Zdonik et al, ACM Computing Surveys, December 1996, Volume 28, Number 4

41. How to improve RDMSs, Nagraj Alur and Judith Davis, Byte magazine, April 1997

42. http://www.w3.org/, The World Wide Web Consortium

43. http://playground.sun.com/pub/ipng/html/ipng-main.html, IPv6 or IPng

44. http://www.w3.org/pub/WWW/Addressing/URL/, Universal Resource Locators

45. The Java Language Specification, Ken Arnold and James Gosling, Addison Wesley, ISBN: 0-201-63451-1

46. http://www.w3.org/pub/WWW/Jigsaw/, The Jigsaw Web server

47. http://www.ibm.com/technology/books/webgate/, Web Gateway Tools, J Cheng and S Malaika, John Wiley 1997, ISBN: 0471-17555-2

48. Web Server Technology, Nancy J. Yeager and Robert E. McGrath, Morgan Kaufmann, 1996, ISBN: 1-55860-376-X

49. http://www.altavista.digital.com/, AltaVista

50. http://www.hotbot.com/, HotBot

51. http://www.webcrawler.com/, WebCrawler

52. http://www.excite.com/, Excite

53. http://www.yahoo.com/, Yahoo

54. http://www5conf.inria.fr/fich_html/papers/P9/Overview.html, Measuring the Web, Tim Bray, WWW5 Conference

55. http://www.aqui.ibm.com/, aqui

56. Automatic generation of Web interfaces to databases, Alistair Dunlop and Mark Paiani, University of Southampton, UK

57. http://www.software.ibm.com/data/net.data/, Net.Data

58. C. Bowman, P. Dantzig, D. Hardy, U. Manber, and M. Schwartz. The Harvest information discovery and access system. WWW'94, Chicago, IL, October 1994.

59. L. Shklar, S. Thatte, H. Marcus, and A. Sheth. The ``InfoHarness'' Information Integration Platform, WWW'94, Chicago, IL, October 1994.
60. http://www.almaden.ibm.com/cs/garlic/homepage.html, The Garlic Project to enable large scale multimedia systems, IBM Almaden

61. http://www-db.stanford.edu/tsimmis/tsimmis.html, The Tsimmis Project to facilitate the rapid integration of structured and unstructured data, Stanford University

62.. West, S. & Norris, M. "Media Engineering: a Guide to Developing Information Products". Wiley, 1997

63. Maurer, H. "Hyper-G now HyperWave: the Next Generation Web Solution". Addison-Wesley, 1996.

64. Rizk, A. & Sutcliffe, D. "Distributed Link Service in the Aquarelle Project". In the Proceedings of the ACM Conference on Hypertext, HT 97, pp 208 - 210, ACM Press (1997)

65. Gronbak, K., Bouvin, N. & Sloth, L. "Designing Dexter-based Hypermedia Services for the World Wide Web". In the Proceedings of the ACM Conference on Hypertext, HT 97, pp 146 - 156, ACM Press (1997)

66. Anderson, K. "Integrating Open Hypermedia Systems with the World Wide Web". In the Proceedings of the ACM Conference on Hypertext, HT 97, pp 157 - 166, ACM Press (1997)

67. Carr, L.A., De Roure, D.C., Hall, W. & Hill, G.J. "The Distributed Link Service: A Tool for Publishers, Authors and Readers". Proceedings of the Fourth International World Wide Web Conference: The Web Revolution, Boston, MA, December 1995.

68. Carr, L.A., Davis, H.C., De Roure, D.C., Hall, W. and Hill, G.J. "Open Information Services". Computer Networks and ISDN Systems, 28, 1027-1036, Elsevier 1996

69. Hitchcock, S., Carr, L., Harris, S., Hey, J. & Hall, W. "Citation Linking: Improving Access to Online Journals" To be published in the Proceedings of the ACM Digital Lirbraries Conference (DL'97), Pittsburgh, USA, July 1997.

70. Lewis, P.H., Davis, H.C., Griffiths, S.R., Hall, W. and Wilkins, R.J. "Media-based Navigation with Generic Links. In Proceedings of the Seventh ACM Conference on Hypertext, ACM, March 1996, pp215-233.

71. Lewis, P.H., Davis, H.C., Dobie, M.R.and Hall, W. "Towards Multimedia Thesaurus Support for Media-based Navigation". In Proceedings of the First International Workshop on Image Databases and Multi-media Search, Amsterdam University Press, August 1996, pp83-90.

72 Goose, S., Dale, J., Hill, G.J., DeRoure, D. & Hall, W. "An Open Framework for Integrating Widely Distributed Hypermedia Resources". In the Proceedings of the IEEE International Conference on Multimedia and Computing Systems (ICMCS'96), Hiroshima, June 1996, pp 364-371, IEEE Press, 1996.

73. Hill, G.J., Hutchings, G.A., James, R., Loades, S., Hale, J. & Hatzopulous, M. "Exploiting Serendipity Amongst Users to Provide SUpport for Hypertext Navigation". In the Proceedings of the ACm Conference on Hypertext (HT'97), Southampton. ACM Press, pp 212 - 214 (1997)