

Singapore Management University Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

8-1996

On integrating existing bibliographic databases and structured databases


Ying LU

Ee Peng LIM

Singapore Management University, eplim@smu.edu.sg

DOI: <https://doi.org/10.1109/CMPSAC.1996.544165>

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [Databases and Information Systems Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

Citation

LU, Ying and LIM, Ee Peng. On integrating existing bibliographic databases and structured databases. (1996). *Proceedings of the 20th Annual International Computer Software and Applications Conference COMPSAC '96, August 21-23, 1996, Seoul, Korea*. 214-219.

Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/999

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

On Integrating Existing Bibliographic Databases and Structured Databases*

Ying Lu, Ee-Peng Lim

{luying,aseplim}@sentosa.sas.ntu.ac.sg
School of Applied Science
Nanyang Technological University
Singapore 639798

Abstract

It is widely accepted that future digital library applications have to be built upon different kinds of database servers to draw from them different forms of data. These data include bibliographic data, text data, multimedia data as well as structured data. In this paper, we address the problem of integrating existing bibliographic and structured databases which reside at different locations in the network. To integrate bibliographic data and structured data, we extended the well-known SQL model to represent bibliographic related attributes and queries. In particular, we have added a new data type to model attributes in the bibliographic database. We have also designed specialized predicates and functions that can be used to formulate queries on bibliographic data alone as well as queries on both bibliographic and structured data. By combining our SQL extensions with the other SQL extensions that model general text data, we believe that a versatile query language layer can be made available to our future digital library application development.

1 Introduction

Traditionally, library systems operate in isolated environment. Although there is an increasing number of library systems being automated and made available on the network, most of these systems are still standalone. On one hand, they do not cooperate in inter-library activities such as *inter-library loans* and *concurrent library searches*. On the other hand, there is also a lack of integration between library systems and other kinds of information resources which exist in the forms of document databases (multimedia or text) and structured databases.

Clearly, to build advanced digital library systems and applications, one has to integrate these different kinds of databases together and to provide value-added search and retrieval services over them[1]. For example, the future digital libraries should allow users

to perform online-catalog searches followed immediately by retrieving the documents associated with selected catalog records within a single environment. Business users should also be allowed to query library databases and their own structured databases in an integrated manner.

In this paper, we focus on the integration of bibliographic databases and structured databases which may reside at different locations in the network. We believe that bibliographic databases maintained by almost all the library systems represent an important source of information that can be shared among library users. To support queries on bibliographic databases as well as queries on both bibliographic and structured relational databases, we have extended the relational model so that remote bibliographic databases can be modeled as tables, and tables can involve attributes from both bibliographic and structured databases. We have also added some bibliographic related predicates/functions to the SQL language[2] so that bibliographic attributes can be manipulated.

1.1 Related Work

Bibliographic data usually demonstrate weak database schema structures (i.e. different attributes can be found in different bibliographic records), and we can classify them as **semi-structured data**. Studies on the integration of general semi-structured data and structured data have been reported in the TSIMMIS project[3, 4], and the T/RDBMS project [5]. They typically focus on integrating relational data with text data which can be represented as labelled graphs, or text data which conforms to the SGML standard. Unfortunately, these proposed schemes do not suit the bibliographic data well since the latter share a more restrictive structure governed by the international standard, i.e. MARC[6]. Hence, the design of a new extended relational model and query language for bibliographic data becomes necessary. Our problem of integrating existing bibliographic and structured databases is very much related to the area of multidatabase research[7]. Multidatabase systems aim to integrate the schemas of multiple existing structured databases together and to

*This work has been supported by the Nanyang Technological University-National Computer Board Memorandum of Understanding Research Grant.

support queries against the combined global schema. In some way, our problem is more complex than that of multidatabase systems because more diversified forms of databases are involved, i.e. bibliographic data and structured data co-exist.

1.2 Motivating Examples

Let **RefDB** and **PubDB** be two structured databases residing at different locations.

Schema of RefDB:

RefTB(RefId, RefType, Title, Author)

Schema of PubDB:

BookTB(BookId, Title, Author, Subject, ISBN, Price, Quantity)

AuthorTB(AuthorId, Name, Address, Telephone, Email)

RefDB is a reference database which contains the reference table about some researcher's interest. **PubDB** is a database owned by a publisher. **BookTB** contains the information of all books published by the publisher. **AuthorTB** contains the particulars of the authors whom the publisher has commissioned. Our example also includes bibliographic databases found in the NTU¹ library and the NUS² library. These two bibliographic databases are in the MARC format. When all these databases are not integrated together, it can be both awkward and time consuming to carry out tasks involving them.

2 Local Bibliographic Database and Query Model

2.1 MARC Standard

MARC³ is a set of standards that describes how cataloging information can be stored or exchanged. It defines a comprehensive set of **fields** that describe library material including books, periodicals, films, maps, sound recordings, etc.. Every field is assigned a **tag** so that it can be manipulated by library softwares. The most popular MARC field tags are listed as follows:

Popular MARC Fields

Tag	Field name
001	Control number
005	Date and time of latest transaction
020	International Standard Book Number (ISBN)
040	Cataloging Source
100	Main entry - personal name
110	Main entry - corporate name
111	Main entry - conference or meeting
245	Title statement
250	Edition statement
300	Physical Description
500	General note
600	Subject added entry - personal name
610	Subject added entry - corporate name
650	Subject added entry - topical heading
700	Added entry - personal name

¹NTU is the abbreviation for Nanyang Technological University.

²NUS is the abbreviation for National University of Singapore.

³Among the various MARC standards, we have chosen to model USMARC closely since it is the most popular MARC variant and there is a trend of other MARC variants converging towards USMARC. For convenience, we shall use MARC and USMARC interchangeably henceforth.

Below shows a bibliographic record formatted in MARC. The record is coded and divided into fields which are variable length strings. Each field is given a tag with a specific meaning. For example, field with tag 100 is a main entry personal name, which is also the author's name. Most fields are subdivided into **subfields** to define individual elements within the fields and to attach more refined meaning to the subfields. Each subfield is assigned a string value together with a **subtag** (e.g. '\$a', '\$b', '\$c', etc.) which is unique within the field. Field 260, which contains publication information, has three subfields: '\$a', '\$b' and '\$c'. They represent the place of the publication, the name of the publisher and the date of the publication respectively.

The book Senn, James A. Information technology in business: principles, practices, and opportunities. Annotated instructor's ed., Englewood Cliffs, N.J.: Prentice Hall, c1995. has the following MARC fields:

```
001 AAS-5906
005 19950106105505.2
050 00 $a HF5548.2 $b .S4366 1995
100 10 $a Senn, James A.
245 10 $a Information technology in business : $b principles,
practices, and opportunities / $c James A. Senn.
250 $a Annotated instructor's ed.
260 0 $a Englewood Cliffs, N.J. : $b Prentice Hall, $c c1995.
300 $a xxiv, 598, 2, 16 p. : $b col. ill. ; $c 26 cm.
650 0 $a Business $x Data processing.
650 0 $a Information storage and retrieval systems $x Business.
650 0 $a Information technology.
650 0 $a Local area networks (Computer networks)
```

Unlike tuples in a relational table, a MARC record usually contains only a subset of all fields defined by MARC. The fields included by a MARC record depend on the type of record (book, serial, film, audio recording, etc.). The same field may appear more than once in one record.

2.2 Z39.50 Information Retrieval Protocol

While MARC is a well-accepted standard to store and exchange bibliographic information, without an information retrieval protocol one still cannot access the existing MARC databases. Recently, a strong effort in standardizing remote library retrievals has resulted in the ANSI Z39.50 protocol[8]. Z39.50 has been implemented at many libraries (including Data Research Public Library, AT&T Research Library and Nanyang Technological University Library) and most library software vendors are quick in adopting it.

Queries to bibliographic databases managed by Z39.50 servers have to be specified as Z39.50 search requests. Z39.50 mandates the query types, namely the Reverse Polish Notation (RPN) query type⁴, to be supported by all Z39.50 servers. In this paper, our discussion will be restricted to only the RPN type queries. The essential features of RPN type queries are:

- *Bib-1 attribute set for specifying search predicates*: Z39.50 is designed to support a variety of bibliographic databases (MARC or non-MARC). Hence, different attribute sets can be

⁴Also known as the type-1 query.

used to specify search predicates within RPN type queries. Among them, the attribute set **Bib-1** has been widely used. Each Bib-1 attribute corresponds to one or more MARC fields. Some Bib-1 attributes and their corresponding MARC fields are given in Appendix A.

- *MARC records as results.*
- *Z39.50 client can query only one Z39.50 server at a time.*
- *Query expressive power of RPN query type:* RPN queries support string related comparisons and the usual boolean operators (AND, OR, NOT). However, all binary algebraic operators, such as join, intersection, union, etc., are not available. Aggregation and nested queries are not supported, either.

3 Integrating Bibliographic Data Model and Relational Model

Our integrated data model is based on an extension of the relational model. Existing bibliographic databases are modeled as relational tables in the integrated data model just like relational tables from other existing relational databases. Local databases and their tables have to be imported into the integrated database before they can be queried. This is done by the `IMPORT DATABASE` and `IMPORT TABLE` commands. For example, to import `RefDB` maintained by a SQL server located at `sentosa.sas.ntu.ac.sg`, we use the command:

```
IMPORT SQL DATABASE RefDB
(IP=SENTOSA.SAS.NTU.AC.SG, port=210)
IMPORT SQL TABLE RefTB@RefDB(RefId int,
RefType int, Title char(60), Author
char(40))
```

3.1 Marc-attribute And MarcString

In our integrated data model, we define **marc-attribute** to be a set of MARC fields sharing the same tag value. Every local MARC database is thus modeled as a bibliographic table (**BIB table**) which consists of marc-attributes modeling the full set of MARC fields. All marc-attributes in a BIB table are of **MarcString** data type described by the following BNF grammar:

```
<MarcString> : ( <tag>, <element>* )
<element> : ( '<subtag> <sub-element>'* )
```

A **MarcString** value consists of a **tag** and a set of **elements**. An element consists of repeating pairs of **subtag** and **sub-element**, which correspond to MARC subfields. Consider the MARC fields of tag 650 in Section 2.1. These MARC fields are represented by a **MarcString** value:

```
(650, ('$a Business' '$x Data processing')
('$a Information storage and retrieval
systems' '$x Business')
('$a Information technology')
('$a Local area networks(Computer
networks')))
```

By including the **MarcString** data type into our integrated data model, tables containing both marc-attributes and structured attributes can also be represented.

A marc-attribute is named `MAttr('attribute-name')` where **attribute-name** is the name assigned by the MARC standard. Marc-attributes can also be referenced by 'MAttr' followed by their tag numbers. For example, the marc-attribute whose tag is 100 can be represented by `MAttr('Main entry - personal name')` or `MAttr100`.

A MARC database is imported as a BIB table in our integrated database. For MARC records that do not have full set of MARC attributes, we assign NULL values to their missing MARC attributes. Unlike importing SQL tables, the importation of BIB tables does not require schema information because all BIB tables share the same schema. For example, we can import NTU library located at host `LIBSERVER.NTU.AC.SG` by:

```
IMPORT BIB TABLE BibTB@NTU_LibDB
(IP=LIBSERVER.NTU.AC.SG, port=210)
```

3.2 Virtual Bibliographic Table

In our integrated data model, a **virtual bibliographic(BIB) table** can be defined upon multiple BIB tables imported directly from local MARC databases. Each of these BIB tables will be called a **member BIB table**. The purpose of defining virtual BIB tables is to facilitate queries to be broadcasted to multiple local BIB tables managed by different servers. In this manner, users do not need to specify separate queries to different servers. Assuming that the NUS library has been imported as `BibTB@NUS_LibDB`, we may define `NTUandNUS_BibTB` as a virtual BIB table consisting of the NTU library and the NUS library information:

```
DEFINE VIRTUAL BIB TABLE NTUandNUS_VBibTB AS
UNION OF BibTB@NTU_LibDB, BibTB@NUS_LibDB
```

The schema of a virtual BIB table is similar to that of any imported BIB table except that it has an extra **location** attribute which is of string data type. It specifies the member BIB table from which a virtual BIB table record is derived. A query on a virtual BIB table is broadcasted to all its member BIB tables, and the results of these identical queries are unioned.

3.3 Bib-1 Attributes

To support Z39.50 queries on the imported BIB tables, a mapping between Bib-1 attributes and marc-attributes is needed. Since Bib-1 attributes can be treated as standard surrogates for marc-attributes, their data types are therefore **MarcString**. Every Bib-1 attribute is named by `BAttr('attribute-name')` where **attribute-name** is the name assigned by the Z39.50 standard. Bib-1 attributes can also be referenced by 'BAttr' followed by their attribute id's. For example, Bib-1 attribute 1003 can be represented by `BAttr('Author')` or `BAttr1003`.

Bib-1 attributes, when used in the search predicates of a query, will be translated into predicates on the underlying marc-attributes connected by OR operation. When a Bib-1 attribute is used in the `SELECT`

clause of a query, it will be replaced by its underlying marc-attributes.

3.4 Alias-attributes

The use of alias-attribute can be best illustrated by the **keyword** concept supported by most bibliographic search engines. A search on keyword corresponds to applying the same search predicate to several bibliographic attributes which are defined to constitute a keyword. Usually, the bibliographic attributes that constitute a keyword include author, title, subject, etc..

Since keyword concept is neither supported as an Bib-1 attribute in the current version of Z39.50 nor included in the MARC standard, our integrated model allows it to be defined as an alias-attribute. For example, to define **keyword** as an alias for **BAttr4** (Title), **BAttr1003** (Author) and **BAttr21**(Subject heading) in **BibTB@NTU_LibDB**, the following **DEFINE ALIAS** statement can be used:
DEFINE ALIAS keyword AS BAttr4, BAttr1003, BAttr21 FOR BibTB@NTU_LibDB

Like Bib-1 attributes, alias-attributes can appear in both **SELECT** and **WHERE** clauses of our queries. The way we handle such queries is similar to those with Bib-1 attributes.

4 Extended Query Language - HarpSQL

As the relational model is extended to represent bibliographic and structured database information, we augment the SQL language with additional features so that queries on the extended relations can be formulated. This extended SQL is known as **HarpSQL**⁵. The complete set of HarpSQL includes the **IMPORT**, **DEFINE VIRTUAL BIB**, and **DEFINE ALIAS** commands described in the previous section. The syntax of **SELECT-PROJECT-JOIN** query in HarpSQL is shown here:

```
SELECT <attributes> FROM <tables>
WHERE <conditions>
```

Apart from handling the usual SQL queries on existing structured databases⁶, HarpSQL allows us to write new queries that involve both bibliographic and structured databases. The extended predicate and functions added by HarpSQL are described below.

4.1 MarcString Containment Predicate

Bibliographic information is modeled as MarcStrings in our integrated data model. Therefore, we introduce a new predicate **Contain()** for comparing an attribute of MarcString data type with a normal string. The design of **Contain()** adheres closely to the suggested Bib-1 query semantics agreed by Z39.50 Implementator Group[8]. **Contain()** has the syntax of: **Contain(attrib, search_term, [search_mode])** where **attrib** specifies the attribute to be searched, and it must be of MarcString data

type. Hence, any marc-attribute, Bib-1 attribute or alias-attribute can be used as **attrib**. **search_term** can be an attribute of string data type or simply a string constant. **search_mode** specifies how the MarcString containment function can be computed. **Contain()** returns **TRUE** when the value of **search_term** is contained in **attrib** according to the desired **search_mode**, and returns **FALSE** otherwise.

search_mode is a quadruple of four sub-modes represented by:
<position,structure,truncation,completeness>

- **position** specifies the location of the search term within the attribute in which it appears. For example, **FIRST_IN_ELEMENT** means that the search term must be the first data in any element of a MarcString.
- **structure** specifies the type of the search term. For example, **IS_PHRASE** requires the search term to be treated as a phrase consisting of one or more words separated by blanks.
- **truncation** specifies whether one or more characters may be omitted in the string matching process. For example, **RIGHT_TRUNC** indicates that the last word of the search term is truncated.
- **completeness** specifies whether the content of the search term represents a complete or incomplete element/sub-element. For example, **COMPLETE_SUBELEMENT** means that only those words in the search term should appear in the sub-element of the MarcString in which the search term appears.

If a search mode is replaced by **NULL**, a default value for that sub-mode will be used. For example, if we want to search the NTU library for the books related to 'distributed database', we may use **Contain(BAttr('Subject heading'), 'distributed database', <NULL, IS_PHRASE, NULL, NULL>)**.

4.2 Sub-element Extraction Function

To allow users to extract specific sub-element values from marc-attributes, we define **Extract()** function as: **Extract(attrib, subtag, num, len)**. **Extract()** returns a normal string concatenating the sub-elements sharing a specified subtag from a marc-attribute. The **subtag** specified can be '\$a', '\$b', etc. **num** (≥ 0) specifies the number of elements from which sub-elements are extracted. If **num** is 0, sub-elements are extracted from all the elements. If **num** is greater than 0, sub-elements are extracted from the first **num** elements of **attrib**. **len** specifies the maximum length of the return string.

For example, marc-attribute with tag 650 (Subject added entry - topic heading) represents subject information. Each element may contain several sub-elements such as \$a for topical heading/place, \$x for general subject subdivisions, etc.. To concatenate all topical heading/place information into a string of at most 256 characters, **Extract(MAttr650, '\$a', 0,**

⁵HarpSQL is chosen because this extended query language has been adopted by an integrated digital library project called HARP[9].

⁶Assuming that the local structured database systems are either SQL-based or support SQL gateways.

256) is used. If only the topical heading/place information from the first three elements are wanted, `Extract(MAttr650, '$a', 3, 256)` is used. The results of applying these two `Extract()` functions on the `MarcString` value given in Section 3.1 are:

```
Extract(MAttr650, '$a', 0, 256) =
'Business; Information storage and retrieval
systems; Information technology; Local area
networks(Computer networks)'
Extract(MAttr650, '$a', 3, 256) =
'Business; Information storage and retrieval
systems; Information technology'
```

4.3 Conversion From MarcString to Normal String

In a query, users may want to convert some marc-attributes into normal strings. For example, the publisher in our motivating example may want to get the subject information of the books from `BookTB@NTU_LibDB` and stores them as normal strings. In this case, `MarcToText(attrib, len)` is the conversion function that can be used. `attrib` represents the marc-attribute to be converted. `len` specifies the maximum length of the return value. Unlike `Extract()`, `MarcToText()` first concatenates all sub-elements of an marc-attribute element into a normal string before it concatenates the strings of all elements together⁷. Given the marc-attribute for tag 650 in Section 3.1, `MarcToText(MAttr650, 256)` returns `'Business, Data processing, Information storage and retrieval systems, Business, Information technology, Local area networks(Computer networks)'`.

4.4 Query Examples Of HarpSQL

In the following, we give some query examples to illustrate the HarpSQL features.

Q1: Retrieve the title statements and author names of the books with the **keyword** phrase 'distributed database' from the NTU library.

```
SELECT MarcToText(MAttr245, 256),
Extract(MAttr100, '$a', 0, 256)
FROM BibTB@NTU_LibDB
WHERE Contain(keyword, 'distributed
database', <ANY_POSITION, IS_PHRASE, NULL,
NULL>)
```

As shown above, HarpSQL can be used to query any remote bibliographic database. In Q1, we assume that **keyword** is defined as an alias for `BAttr1003(Author)`, `BAttr4(Title)` and `BAttr21(Subject heading)`. The search term 'distributed database' is used as a phrase and can appear in any position in the three Bib-1 attributes. To evaluate the query, the `Contain()` predicate involving **keyword** is replaced by a disjunction of `Contain()` predicates involving the three Bib-1 attributes.

```
Contain(BAttr1003, 'distributed database',
<ANY_POSITION, IS_PHRASE, NULL, NULL>) OR
```

⁷Two strings are concatenated with a comma between them. All subtags are discarded.

```
Contain(BAttr4, 'distributed database',
<ANY_POSITION, IS_PHRASE, NULL, NULL>) OR
Contain(BAttr21, 'distributed database',
<ANY_POSITION, IS_PHRASE, NULL, NULL>)
```

Q2: Retrieve the title, subject and location information of the books authored by 'John Smith' from the NTU and the NUS libraries.

```
SELECT BAttr4, BAttr21, Location
FROM NTUandNUS_VBibTB
WHERE Contain(BAttr1003, 'John Smith',
<NULL, IS_NAME, NULL, NULL>)
```

Here, instead of writing two identical queries to the two bibliographic servers, the user simply queries the virtual BIB table `NTUandNUS_VBibTB`.

Q3: Retrieve from the NTU library the titles, authors and subjects of books found in `RefTB`.

```
SELECT a.BAttr4, a.BAttr1003, a.BAttr21
FROM BibTB@NTU_DB: a, RefTB@RefDB: b
WHERE Contain(a.BAttr4, b.Title,
<FIRST_IN_SUBFIELD, IS_PHRASE, NULL, NULL>)
AND Contain(a.BAttr1003, b.Author, <NULL,
IS_NAME, NULL, NULL>)
```

To use the information in `RefTB` to search the BIB table in the NTU library, HarpSQL allows SQL and BIB tables to be joined in one single query. In the query, **a** and **b** are aliases for `BibTB@NTU_DB` and `RefTB@RefDB` respectively. The join between the two tables is expressed by conjunction of two `Contain()` predicates. In the predicates, `b.Title` is treated as a phrase and it must be the first data in any of the sub-element of `a.BAttr4`. `b.Author` is treated as a person name.

Q4: Retrieve the author names, titles and subjects of the books written by the authors whose names are listed in `AuthorTB` from the NTU library and store the result as a HarpSQL table in `MyDB`. Here, `MyDB` is a HarpSQL database that is owned by the digital library user, and is designated to store temporary results.

```
CREATE ResultTB@MyDB (Name CHAR(60) NOT
NULL, Title MarcString NOT NULL, Subject
MarcString);
INSERT INTO ResultTB@MyDB
SELECT b.Name, a.MAttr245, a.MAttr650,
FROM BibTB@NTU_DB: a, AuthorTB@RefDB: b
WHERE Contain(a.BAttr1003, b.Name, <NULL,
IS_NAME, NULL, NULL>)
```

As Q3, this query involves both a BIB table and a SQL table. The `CREATE` statement is used to create a new HarpSQL table `ResultTB@MyDB`. The latter consists of both bibliographic and structured attributes for storing the query result of the `SELECT-PROJECT-JOIN` statement.

5 Conclusions

In this paper, we proposed extensions to relational model and SQL language to represent information found in remote bibliographic databases, specifically those accessible through the well-accepted Z39.50 information retrieval protocol. By modeling these bibliographic databases as relational tables, the end-users and application developers can now easily comprehend these bibliographic databases similar to the standard SQL databases. The extended model also serves to integrate bibliographic databases and relational databases in a loosely coupled manner.

Our proposed SQL extension, i.e. HarpSQL, supports importation of remote SQL and MARC databases. It also supports queries that involve one or more bibliographic databases, as well as queries that involve both bibliographic and relational databases. To achieve the integration of two query paradigms, we have proposed new predicates and functions. Currently, our extended relational model and HarpSQL are being prototyped as part of an integrated digital library project called **Harp**[9].

References

- [1] ACM. *Communications of the ACM*, volume 38. April 1995.
- [2] S.J. Cannan and G.A.M. Otten. *SQL - The Standard Handbook Based On the New SQL Standard (ISO 9075:1992(E))*. McGraw-Hill Book Company, 1993.
- [3] S. Chawathe, etc.. The TSIMMIS project: Integration of Heterogeneous Information Sources. In *IPSJ Conference*, Tokyo, 1994.
- [4] D. Quass, etc.. Querying Semistructured Heterogeneous Information. In *Proceedings of the Conf. on Deductive and Object-Oriented Databases*, Singapore, 1995.
- [5] G.E. Blake, etc.. Text/Relational Database Management Systems: Overview and Proposed SQL Extensions. Technical Report 95-25, UW Centre for the New OED and Text Research, University of Waterloo, 1995.
- [6] W. Crawford. *MARC for Library Use: Understanding the USMARC Formats*. Knowledge Industry Publications, Inc., 1984.
- [7] A.P. Sheth and J.A. Larson. Federated database Systems for Managing Distributed Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3), September 1990.
- [8] ANSI Z39.50: Information Retrieval Service and Protocol, 1992.
- [9] E-P. Lim, S-Y. Cheng, and B-S. Lee. Harp: An Integrated Digital Library. In *Proceedings of International Symposium on Digital Libraries*, Ibaraki, Japan, August 1995.

Appendix A : Mapping Between Bib-1 and MARC Attributes

Bib-1 Id	Name	MARC Tag(s)
62	Abstract	520
1003	Author	100,110,111,400,410,411,700,710,711,800,810,800
13	Dewey classification	082
16	LC call number	050
12	Local number	001,035
31	Date of publication	008,260,046,533
32	Date of acquisition	541
7	ISBN	020
3	Conference name	111,411,611,711,811
1	Personal name	100,400,600,700,800
63	Note	5xx
21	Subject heading	600,610,611,630,650,651,653,654,655,656,657,69X
27	LC subject heading	600,610,611,630,650,651
4	Title	130,21X-24X,400,410,440,490,600,610,611,700,710,711,730,740,800,810,811,830,840
5	Title series	400,410,411,440,490,800,810,811,830,840