ISO/IEC JTC 1/SC 32 N 0273

Date: 1999-05-13

REPLACES: --

ISO/IEC JTC 1/SC 32

Data Management and Interchange

Secretariat: United States of America (ANSI)

Administered by Pacific Northwest National Laboratory on behalf of ANSI

DOCUMENT TYPE	Other (Open)
TITLE	Z39.50 and RDA : Two Standards for Access to Remotely Held Information
SOURCE	National Body - Canada
PROJECT NUMBER	
STATUS	Canadian Contribution to WG2 and WG5
REFERENCES	
ACTION ID.	FYI
REQUESTED ACTION	Review by WG 2 and WG 5
DUE DATE	
Number of Pages	53
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 901 D Street, SW., Suite 900, Washington, DC, 20024-2115, United States of America

Telephone: +1 703 575 2114; Facsimile; +1 703 681 9180; E-mail: MannD@battelle.org

*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

CAC/JTC1/SC32 C0057

Title: Z39.50 and RDA : Two Standards for Access to Remotely Held Information

Source: CAC/JTC1/SC32, Canada

Status: Canadian Contribution to WG2 and WG5

Author: Mazhar Islam

Date: 12 May 1999

CAC/SC32/0057

Z39.50 and RDA : Two Standards for Access to Remotely Held Information

Work performed for CONSULTING AND AUDIT CANADA, under Contract No. 510-8160, Project No. 510-2180, dated, May 20, 1997. Vendor No. 01Z5G6.

March 31, 1999

Mazhar Islam Consulting
8 Leona Avenue
Nepean, Ontario, K2E 5H8
Phone : (613) 224-2016

Z39.50 and RDA : Two Standards for Access to Remotely Held Information

CONTENTS

ABSTRACT

The ANSI/NISO Z39.50 (Information Retrieval) and ISO/IEC 9579 (Remote Database Access) protocols are examined for their similarities and differences. The ANSI protocol is designed to read/copy/retrieve records from database services that offer a flat-file interface to external users. In its successive versions the Z39.50 protocol has become richer in functions and options but without undergoing significant change in its basic design. The ISO protocol was designed initially to provide full function remote access (read/write/update) to all types of databases through various specializations of the RDA Generic protocol. However, only one specialization was ever defined, namely, for the database language SQL-1992. The latest version of RDA has been greatly simplified in design and its scope limited to providing support for SQL-connections between SQL-clients and SQL-servers. Initially, both the ISO and the ANSI protocols were designed to be mapped on to the OSI Presentation service, but their latest versions may operate directly over the TCP/IP transport service. Various enhancements to the two protocols planned or already developed are discussed. In conclusion some remarks are offered on their future prospects.

1. Scope of this report

The scope of this report is indicated in the contract work statement as to "Prepare and deliver a presentation describing the current and draft versions of the RDA standard and comparing it to the Z39.50 standard. The presentation will cover design and technical aspects addressing a technical audience". [1]

The "current version" of the RDA standard is to be understood as the three part standard ISO/IEC 9579-1:1993 [9] and ISO/IEC 9579-2:1997 [10]and ISO/IEC 9579-3:1995 [11]. The "draft version" of RDA is what has been informally known as RDA3 - a greatly simplified and redesigned protocol specified in a single ISO standard document bearing the number ISO/IEC 9579 :1999. The final text of RDA3 was balloted and approved in March 1999.

There are no official version numbers of the RDA standard. The original draft of RDA SQL Specialization was based on the use of SQL89 (ISO 9075-1989), but when the latter was replaced by SQL92 (ISO 9075-1992) the final text of RDA standard ISO 9579-2-1997) was modified accordingly. Informally, RDA1 and RDA2 refer respectively RDA SQL Specializations based on SQL89 and SQL92.

For the purpose of this report both Versions 2 and 3 of Z39.50, an ANSI/NISO standard, are considered. There is no need to consider the obsolete original version Z39.50-1988 sometimes referred to as Version 1. Version 3 is a superset of Version 2, ie, the Z39.50-1992 [2]. It is worth noting that the Z39.50-1995 [3] has now been adopted as an ISO standard with the ISO Ref. No. ISO/IEC 23950. Thus ANSI/NISO Z39.50-1995 and ISO 23950 have identical text.

2. Versions compared

A detailed comparison of the two standards must take into account the fact that each has undergone substantial developments in successive versions. Z39.50 has developed towards more functions and more options but without any drastic change in the design philosophy or the basic architecture; the aim has been to retain backward and forward compatibility between versions.

RDA has developed in quite the opposite direction. The latest version is a completely redesigned protocol which is not backward compatible with the earlier versions. The goal of capturing the common aspects of all remote database access applications in a Generic part of the standard has been abandoned in RDA3 together with the generic/specific structural distinction in favour of the more modest and pragmatic goal of supporting the SQL-connection between SQL-clients and SQL-servers in relational database systems. By lowering its sights RDA3 has gained considerably in simplicity and completeness of specification. A single document (ISO 9579-1999) of no great size includes the complete specification including the specification of a language independent Application Programming Interface (API).

In view of the similarities of their basic architecture Z39.50-1992 and RDA2 (ISO 9579-1:1993 and ISO 9579-2:1997) may be compared in a relatively straightforward manner. This is what I have attempted in this report. Both were positioned in the OSI upper layer architecture in the same way, and both assumed the use of ACSE, OSI lower layer services, and ASN.1 encoding.

This leaves RDA3 to be compared with Z39.50-1995, even though in these latest versions they differ to a far greater extent than ever. The latest versions of the two protocols are positioned in a 5-layer communication architecture, and may be viewed as operating directly on top of the TCP/IP Transport service, even though both protocols are written as transport service independent in the sense that both may be mapped on to other Transport services as well.

3. An Overview of Z39.50

Z39.50 was conceived as an “information retrieval” protocol with no concern for transaction processing or database update; the latest version retains this limitation of functional scope. In contrast, RDA with its overt database orientation was designed from the outset as a full feature database access protocol, providing facilities for read, write, and update including support for transaction coordination of remote databases.

Work on both RDA an Z39.50 was initiated in the 1980s when expectations were high on the future of the Open Systems Interconnection (OSI) program, and both were conceived as OSI application layer protocols. But at this point in time the only lower-layer services that they require is a reliable full-duplex byte stream transport provider such as TCP/IP. A TCP port number for Z39.50 is registered, and there is a Request for Comment (RFC) that specifies how to use Z39.50 over TCP [5]. Likewise a TCP port number has also been assigned to RDA3 and initial implementations will certainly run directly on top of TCP/IP.

The Z39.50 is a synchronous, connection-oriented protocol that maintains state. The protocol defines interactions between two protocol machines only. The Z39.50 standard does not specify an Applications Programming Interface (API) to the services of the protocol on either the client or the server. It deals only with the interactions between the client and server protocol machines. In addition, Z39.50 does not address any of the issues involved in user interfaces that the client may present or any of the issues involved in database.

The basic application model of Z39.50 is as follows: A database server is associated with one or more databases containing “records”. Associated with each database are a set of access points (indices) that can be used for searching the database for records. This is a more human user oriented view of a database and database search than one finds in the RDA standard.

One of the basic Z39.50 functions allows the client to transmit a SEARCH request to the server. A search produces a set of records, called a "result set", that are maintained on the server; the result of a search is a report back to the client of the number of records comprising the result set. Result sets can be combined or further restricted by subsequent searches. This is quite different from SQL servers, which do not employ result sets.

Records from the result set can be subsequently retrieved by the client using PRESENT requests. The PRESENT request offers elaborate options for controlling the contents and format of the records that are returned. The PRESENT request indicates specifically which records from the result set are to be retrieved.

The standard is silent as to whether the result set is materialized or maintained as a set of record pointers, and as to how the result set is affected by the database updates that may be taking place at the server.

Facilities are provided for managing buffer space in the presence of very large records and also for transferring very large numbers of records from server to client without the need for repeated PRESENT requests and responses involving many round-trip interactions between client and server.

Z39.50 also contains functions for search management. For example, a server can provide progress reports for an active search, or can ask the client for authorization to continue a resource intensive search; a client can abort an active search. The report for search completion can also return supplementary information such as how many records matched individual component terms in a search.

Z39.50 V3 contains facilities for managing result sets, for sorting result sets, for browsing the values of access points associated with a database, for opening and closing connections, and also a general mechanism called **Extended Services**, which is essentially an asynchronous remote procedure call mechanism that the client can use to invoke services on the server, optionally making reference to the contents of a result set as a parameter. Extended services were originally intended as a means of saving result sets across sessions, queuing them for print or electronic mail processing at the server, or for registering and managing queries that would be executed periodically on the server.

The Z39.50 standard also defines the following:

A query language for specifying searches, which in turn builds upon registered definitions for attribute sets that specify the names of access points.

Various record formats that can be used for transferring records from the server to the client, including both application domain specific formats like MARC for bibliographic data and a complex, general purpose syntax called Generalized Record Syntax One (GRS-1).

A language for describing how to construct records that are to be transferred from a result set back to the client.

A facility called EXPLAIN which allows clients to obtain a wide range of information from a server about what databases are available, what access points are supported in each database, and the like. EXPLAIN is modeled as a special-purpose database which is searched using standard Z39.50 queries; the standard specifies the detailed structure of the records that can be retrieved from this database. EXPLAIN is intended to permit the development of clients that are to some extent dynamically self-configuring as they encounter various servers.

Z39.50 makes extensive use of registries for various types of objects, such as attribute sets used in queries and record syntaxes used in present requests. These are referred to via Object Identifiers which are used as parameters in the various protocol requests and responses that move between client and server. Some initial Object Identifiers are assigned by the standard; assignment of object identifiers on an ongoing basis is handled by the Z39.50 Maintenance Agency.

In view of the numerous optional features of Z39.50, particularly in V3, Z39.50 implementations require the definition of "**profiles".** This is reminiscent of the profile definitions required for the implementation of RDA2. Needless to say this introduces an extra layer of complexity and associated problems.

NOTE *: A more detailed description of the various services of Z39.50 are described in Annex : A*

Z39.50 Version 2, was fairly widely implemented particularly in the libraries and information service sectors. Version 3 was much more ambitious than version 2 and included everything that anyone participating in the Z39.50 Implementor's Group (or ZIG) wanted. By this time, however, the implementor's group was much larger and more diverse, including major information services providers like Lexis/Nexis, Dialog, and Chemical Abstracts, as well as the traditional constituencies, namely, the libraries. These new participants brought with them a vast range of new requirements and sometimes a fundamentally different view of the role of standards and interoperability. [6]

The resulting version 3, balloted in 1995, contained a number of important incremental changes like **Segmentation** , **Sorting**, and access point **Browsing**; it also introduced the **Explain** database. But version 3 also introduced very complex features like **Extended Services** and the **Generalized Record Syntax**, which were major departures from previous protocol versions, and which were to raise more fundamental questions about the appropriate scope of the Z39.50 protocol and about the nature of interoperability one might expect from conformant implementations.

Version 3 was much larger than version 2, about 160 pages as opposed to about 40 for the earlier version. In version 2 very little was optional, while the vast majority of the new functionality and changes in version 3 were optional. The actual set of changes necessary to move from a version 2 implementation to a minimal conformant version 3 implementation are not very large, with much of the work for a server being to politely decline to perform various optional functions.

An additional reason for the bulk and apparent complexity of version 3 was that it, in fact, included version 2. Version 3 was designed as a superset of version 2, which incorporated the ability to fall back to the older version 2 specification if the parties involved did not support version 3 for the sake of backwards compatibility with the existing base of implementations. This seemed like a good idea at the time. But in hindsight, it is not clear that the amount of confusion and complexity it created in the standard was really worthwhile.[6]

Version 3 of the standard explicitly recognized the TCP/IP Internet environment in an appendix but also contained carefully crafted language which still permitted Z39.50 to be viewed as an OSI protocol by those who wished to do so.

4. An overview of RDA2

The RDA standard was intended to support interworking between an application program in an open system and a database management system in a remote open system. The RDA service enables a user to use the same front end to access different database systems.

Like the Z39.50 protocol the RDA protocol is connection-oriented and maintains the state of the interaction between the client and the server machine. The protocol defines interactions between two machines only but does not specify an Applications Programming Interface (API) to the services of the protocol on either the client or the server. It deals only with the interactions between the client and server machines. In addition, it does not address any of the issues involved in user interfaces that the client may present or any of the issues involved in database.

The Generic RDA standard (ISO/IEC 9579-1:1993) defines the common aspects of a class of RDA applications, and a specialization standard defines an optimization of the generic standard for a particular type of database system and query language. Only the SQL specialization standard (ISO/IEC 9579-2:1993) has been defined.

RDA uses a client-server model. Typically, an RDA client is an application program running in an intelligent workstation, and an RDA server is a remote database server. Both the RDA client and the RDA server are users of the RDA service.

An RDA server may have multiple dialogues with its RDA clients. It uses the objects in the Dialogue State Model to keep track of its dialogues. Objects in the dialogue state model consist of dialogue entities, opened data resource entities, defined DBL entities, and operation entities. These entities are manipulated by the RDA server.

The RDA services are divided into five groups. The Dialogue Management service allows a client to establish or terminate a dialogue with a server. The Transaction Management service allows transactions to be handled within a dialogue. The Control service allows a client to query the database server for the status of a particular outstanding operation, and to cancel an outstanding operation. The Resource Handling service is used by a client to make available data resources on a remote server. The DBL service is used by a client to execute DBL statements.

The server rules specify constraints on a RDA server when it is processing an RDA request. There are three types of server rules. They are the entity modification rules, the result rules, and the error rules. For example, the result rules specify constraints on the result responses from a server.

An RDA specialization standard defines those aspects that are specific to a particular type of database systems. For example, it defines the format and meaning of any data items that are unspecified by the generic standard. It may define additional entities in the Dialogue State Model, and additional attributes of the existing entities.

***NOTE* :** *For the upper layer architecture of RDA2 see Annex : B1*

5. A Comparison of the RDA2 and Z39.50 V2

The basic application model of RDA is that of a single client accessing a single remote database server. Units of information retrieved depend on the application, the database model, and the query language used.

The basic application model of Z39.50 is also that of a single client ("originator") accessing a single remote server ("target"). However, the server is deemed to offer a flat file view of information to remote clients. The file ("database") is assumed to contain only one record type and only one version of the same. The unit of information retrieved is a predefined "record", that consists of a set of data elements.

Subsets of the predefined record type may be defined outside the Z39.50 standard, and negotiated at the time of initialization. Access to the file and its contents (i.e., records) are effected through the use of a predefined set of "attributes". The Z39.50 standard has defined a set of attributes for bibliographic applications. Other attribute-sets for other types of applications may be defined and registered.

In view of the above differences the services provided by the two protocols can be compared only very roughly even where they are seemingly equivalent . However, certain services are included in one or the other protocol but not in both. In the following table I have tried to bring this out.

RDA2 Services	**Z39.50-1992 Services**
R-Initialize	IR-Init
R-Terminate	IR-abort
	IR-release
R-Open	
R-Close	
R-Cancel	
R-Status	
R-ExecuteDBL	IR-search
	IR-present
	IR-delete-result-set
R-Define DBL	
R-Invoke DBL	
R-Drop DBL	
R-Begin Transaction	
R-Commit	
	IR-access-control
	IR-resource-control
	IR-trigger-resource-control
	IR-resource-report

The two protocols differ most significantly in two areas, namely, transaction management and resource control. RDA supports transaction management but Z39.50 does not. On the other hand, Z39.50 supports resource control and access control but RDA does not. Other significant similarities and differences are high lighted below.

Protocol structure :
Both protocols fit within the OSI application layer structure and require services provided by ACSE and the Presentation layer.

Each RDA interactions between the client and the remote server is defined to have the uniform structure of operation/result/error. Thus, RDA can operate in both asynchronous and synchronous modes.

Z39.50 is a synchronous protocol and can not operate in asynchronous mode without protocol modification. Only the Generic RDA protocol is free of database semantics to the greatest extent possible. This permits the Generic RDA to be specialized for any type of database environment without affecting the protocol.

Semantics of the Z39.50 database are reflected in the structure of Z39.50 protocol. For example, several parameters in the Z39.50 Search and Z39.50 - Present services are concerned with fitting integral number of Z39.50 "records" in the APDUs returned to the client. Rules for processing these parameters by the server are defined in the Z39.50 Standard.

Transaction management and database consistency is fully within the functional scope of RDA SQL Specialization. In the simple environment of one client/one server, the transaction management capability of the SQL server is used. In the more complex environment of multiple servers, RDA operates under the OSI TP which provides the 2-phase commit and rollback functions.

Z39.50 has no concept of transaction management or database update. It has to be significantly enhanced to provide update and transaction management functions.

Z39.50 protocol provides a parameter for the user to indicate the maximum message size (total number of octets of all the records in the message) acceptable to the client. RDA has no facility to control the volume of data returned to the client as a result of processing a single query.

Query Language used

Each protocol is a passive carrier of messages from the client to the database server. The syntax of these messages are governed by the information model and the query language used.

The Z39.50 RPN-Query is essentially a syntactic device; it is adequate for search queries whose semantics require little more than a set of global attributes and some logical connectives. Attributes of a Z39.50 Query belong to a single record type (e.g., bibliographic record). Thus, a Z39.50 Query can not include attributes of two or more record types.

The Z39.50 query processing is much like evaluating a filter. A filter is a construct specifying assertions about the presence or value of certain attributes (e.g., of a record). A filter uses the logical operators AND, OR, and NOT. An attribute-value-assertion is evaluated by matching for Equality, Ordering, Substring, Greater-or-Equal, Less-or-Equal, Present, etc. Filters can be nested to any depth.

Z39.50 uses a similar approach to query formulation and evaluation. Attributes of the record type are combined by using the logical operators AND, OR, and AND-NOT. Attribute-value-assertions are matched for Equality, etc. Operators and operands are strung together in parentheses-free Reverse Polish Notation (RPN).

RDA SQL Specialization uses ISO SQL (ISO 9075:1989, ISO 9075-1992) data manipulation language to formulate queries making use of relational concepts defined in the database schema.

A single RDA SQL Query can search, update and retrieve information contained in any number of tables. Tables can be manipulated to produce new tables by cartesian products, unions, intersections, joins on matching columns, projections on given columns, etc.

SQL has powerful constructs for expressing conditions, performing arithmetic and summary operations, predicates for comparison and string operations, features to partition tables by groups, etc, all in a declarative or non-procedural fashion and completely transparent to the communication protocol.

As the most commonly used database language, standard SQL continues to grow in power and sophistication thus making it usable in an ever widening circle of applications .This assures the widening scope and applicability of RDA based on SQL. There is hardly any such prospect for RPN-Query and the protocols that use these type of queries.

Z39.50 is designed to retrieve whole records or whole documents. In Z39.50 a clear distinction is maintained between "attributes" and "content”. Attributes used in search queries of Z39.50 are "global", that is, they are attributes of the whole Z39.50 record . Attributes are managed independently of content and can be read and changed without affecting "content". Z39.50 has no notion of index terms for indexing content of the Z39.50 record .

It is difficult to see how conditional clauses, predicates, special functions ,etc, needed in content search and retrieval can be expressed in the RPN-Query. Use of a different query language (eg, SQL) in Z39.50 may be very awkward or well-nigh impossible in view of the close coupling of the database semantics with the Z39.50 protocol design. Even so, attempts are now underway to extend Z39.50 so as to be able to use the power of the SQL query language (see section 9 below).

6. An Overview of RDA3 (ISO/IEC 9579:1999)

In RDA3 the ambitious aim of supporting remote access to all types of databases is abandoned in favour of supporting access to just SQL databases. The notion of specializing the Generic RDA for specific database environments is dropped. Instead, the scope of the protocol is limited to providing the communication support for the “SQL-connection” between SQL-clients and SQL-servers as defined in the ISO 9075:1992. The major differences between the latest and the previous versions of RDA can be briefly stated as follows :

RDA old version (RDA2)	**RDA new version (RDA3)**
Most General	Most specific (SQL)
Numerous options	Very few options,
Profile definition required	No Profile definition required
Any database language	Only ISO SQL
7-layer architecture	5-layer architecture
ACSE, Presentation required	ACSE, Presentation not required
No API defined	One API defined
No Transport mapping defined	Mappings to TCP and TLS defined

The result is a standard that is fully specified in just one ISO standards document (ie, ISO 9579:1999) instead of three (ie, ISO 9579-1, ISO 9579-2, ISO 9579-3) for the previous versions.

This impressive economy and completeness in the specification of the protocol is achieved through the adoption of concepts, facilities, and notational conventions of ISO SQL in general and SQL/CLI in particular, adoption of a simple 5-layer communication architecture, definition of a default encoding of RDA protocol messages, and finally the definition of a language independent API. Furthermore, by providing the specification of a Location Server and a Support Server the RDA3 standard has paved the way for RDA support of heterogeneous database interoperation.

The RDA3 is a lightweight protocol in that it is not burdened with the task of maintaining a large number states. Much of the work of maintaining the states of the SQL-connection or SQL transactions is pushed up into the realm of SQL implementation. In RDA3 the protocol structure is greatly simplified by adopting a single message format for all the request protocol messages and likewise for all the response messages. The protocol elements and their constituent parts are encoded for transmission in accordance with either the ASN.1 PER encoding or the Default Encoding which is fully defined in the RDA3 standard.

NOTE : *For more details on the RDA3 see Annex D.*

7. A comparison of Z39.50-1995 and RDA3 (ISO 9579-1999)

A comparison of these two protocols is much trickier than a comparison between their respective earlier versions. Assuming that for all practical purposes, both RDA3 and Z39.50 V3 operate directly on top of the TCP/IP transport provider, both the protocols are positioned in a simplified 5-layer architecture.

RDA3 is able to take advantage of this simpler architecture to a far greater extent than Z39.50. The basic structure of Z39.50 V3 protocol remains unaltered even when it is meant to be run directly on top of TCP/IP. The type of difficulties this creates for Z39.50 implementors is well described in Ref [5]. Much of these difficulties relate to the production of valid encodings for transfer via the TCP/IP without the benefits of a Presentation service.

A significant difference between the RDA3 and the Z39.50 V3 protocols is that while one is designed to support a very specific type of relationship between a database client and a database server, namely, "SQL-connection", the other protocol continues to operate within a general purpose client/server association (Z-Association within an A-Association). Consequently Z39.50 has to maintain a much larger number of states than RDA3.

RDA's close relationship with SQL makes it possible for the client side to request performance of functions on the database side which in Z39.50 would require explicit protocol mechanisms. Consider for example, use of stored procedures via SQL PSM in RDA versus Extended Services in Z39.50 V3. Consider another example : SORT, EXPLAIN in Z39.50; in RDA these would not require any protocol mechanism, ie, exchange of protocol messages.

Transaction coordination is completely out of scope for Z39.50 whereas RDA provides facilities for ensuring that Service User operations leave the SQL-environment in a consistent state. Three types of RDA Transaction Coordination are supported in RDA3 :
- *encompassing*, transaction coordination is the responsibility of some external transaction management component.
- *one-phase*.
- *two-phase*.

Security facilities in both RDA and Z39.50 are currently quite rudimentary. In RDA3 only one type of authentication, namely, *password* is supported. The User Name is authenticated using a password carried in the Authentication field. However, an amendment to the RDA3 standard is intended to provide more comprehensive authentication mechanisms.
For more details see Annex D.

8. Ongoing Work on Z39.50 [6]

Since the adoption of version 3 in the USA in 1995, developments have been proceeding in a number of directions. The independent international text of ISO 10162/10163 (Search and Retrieve Service/Protocol, a subset of Z3950-1992) has been superseded by the international adoption of the NISO Z39.50-1995 text, meaning that there is now only one standards document to work with, rather than multiple documents describing what is hopefully the same protocol.

International participation in the Z39.50 Implementor's group (or ZIG) has grown substantially, with a strong representation from Europe but also growing interest from Australia. The ZIG has been meeting once a year for the past few years as an unofficial advisory group to the Maintenance Agency for the ongoing development of the standard.

Presumably future versions of the standard will be balloted within NISO, and perhaps within ISO internationally as well, although they are not being developed within the normal standards development processes for these organizations. With the growth of international participation, there has been an increased focus on issues such as support of multiple character sets and languages.

Various groups have been developing Z39.50 profiles. The maintenance agency keeps a list of these, but the process by which they are approved and subsequently maintained remains somewhat unclear. Profiles are basically customizations of the standard to particular communities of implementors with common applications requirements. A profile may include a whole range of agreements: for example, agreements to use or not to use specific optional version 3 features; agreements on particular attribute sets and record syntaxes to be used (including perhaps the definition and registry of new attribute sets

and/or record syntaxes to support the community in question); and even agreements on what extended services will be
used (including, again, definitions of new extended services that the profile's community may want to use).

Often it is doubtful how much meaningful interoperability will be possible between one Z39.50 implementation that is built according to a given profile and another which is not aware of the specific profile. Examples of profile work include GILS, the Government Information Locator System; the Museum Interchange Profile being developed by the Computer Interchange of Museum Information (CIMI) group; the Digital Collections profile under development by the Library of Congress; the (revised) WAIS profile; profiles for applications involving remote sensing and geospatial data, and a cataloging profile under development by the National Library of Australia.

The development of profiles signifies the fragmentation of the Z39.50 implementor community into more specialized and potentially insular sub-communities. It is no doubt a response to the interoperability problems raised by the vast number of optional or incompletely specified features in version 3 of the standard.

Finally, one can also view profile development within the Z39.50 community as a response to the lack of other well-defined processes for establishing standards for attribute sets and record interchange syntaxes to support various semantic classes of information resources (such as museum information); these are developed as Z39.50 profiles rather than separate parallel standards that are used in conjunction with Z39.50.

There is work underway on linkages between Z39.50 and various other standards activities. URLs have been defined for Z39.50 database queries, for example. There is an active effort to incorporate SQL as an alternative query language with Z39.50 search requests, although a complete definition of the requirements, limitations, and expected benefits of such an integration remain somewhat unclear. People are beginning to think about how Z39.50 and CORBA might inter-relate.

And, of course, there is discussion about the possible development of version 4 of the standard, about what principles might guide the development of such a version, and what requirements might shape it. At present no consensus exists on such guidelines, and there is no firm commitment or timetable for a new version of the standard.

There does seem to be a general feeling that it will be important to simplify and streamline future versions of the standard; that it is important to more rigorously separate semantic definitions that are specific to certain classes of databases, such as attribute sets and record syntaxes, from general protocol mechanisms that are relevant of all databases; and that the elaborate backwards compatibility requirements that characterized the transition from Z39.50-1992 to Z39.50-1995 may not be necessary in future.

9. Proposed SQL extension to Z39.50 [7, 8]

Recent work in Australia has pointed out the advantages of extending Z39.50 V.3 to include the database language SQL as an alternative to the Query Type-1 (based on the RPN syntax). The SQL enhanced Z39.50 is provisionally labeled Z39.50/SQL+. It has been proposed that SQL extensions to the ANSI/NISO Z39.50-1995 be included in Version 4 of the standard.

This move to incorporate SQL in Z39.50 is reminiscent of the SQL Specialization of Generic RDA, and the FIPS proposal to specify general purpose SQL External Repository Interface (SQL/ERI) profiles for non-SQL data repositories. The proposed ERI profiles were to specify how a subset of SQL standard could be used to provide limited SQL access to legacy databases, or to support SQL gateways to specialized data managers. It is interesting to note that the Z39.50/SQL+ makes use of certain RDA2 concepts (eg, R-InvokeDBL-Result).

Z39.50/SQL+ can be seen as an extension of the existing Z39.50-1995 (Version 3) protocol, uniting the advantages of SQL's query language and export syntax with the information retrieval services of Z39.50. The SQL extension provides a standardized way of specifying complex structured queries which otherwise are not possible with the existing Z39.50 query types. In addition, result sets (which may contain complex data types) may be returned in a generic record syntax without having to tag each individual field of each record and without being tied to a pre-defined schema.

Z39.50/SQL+ would provide the Z39.50 client with the full flexibility and query power of SQL. Z39.5O/SQL+ clients would be able to specify complex queries, either by using SQL or one of its derivatives, such as Query-by-Example (QBE). Queries can be formulated on single (virtual) tables, as in now the case, or multiple tables supporting cartesian products, unions, intersections, joins on matching columns, and projections on given columns. Queries can also be formulated using powerful constructs for expressing conditions, performing aggregate and comparison operations. partitioning tables into groups and much more.

It is claimed that the proposed Z39.5O/SQL+ standard would remain platform and database independent. Indeed, the supporting database may be an RDBMS, OODBMS, O-RDBMS or even a text database for that matter. As is now the case with Z39.50, the onus is on the server to map the Z39.50 PDUs to the applicable database calls.

Z39.50/SQL+, like Open Database Connectivity (ODBC) and Remote Database Access (RDA) Protocol Part 2 - SQL Specialization, uses SQL as its standard way of expressing complex queries on structured data. However, it should be noted that the functionality of Z39.50/SQL+ is conceptually different from that of ODBC and RDA. Both of these simply convey SQL statements (both retrievals and updates) and related control operations (including concurrency control and synchronization) to the database management system. In contrast, Z39.50/SQL+ would have a far greater role in the management and maintenance of states within and across communication sessions.

Z39.5OISQL+ would provide additional querying and retrieval power to existing Z39.50 implementors as well as expand the usage capabilities of Z39.50 away from the library-type communities, which are mostly text based, to a much broader community employing a wider class of database as part of their information management resource base.

For more details on Z39.50/SQL+ see Annex C

10. Ongoing work on RDA

Work is currently underway in several areas to enhance the new RDA3 standard. Of these the following are the most important :

- Support for SQL3/CLI
- Secure RDA
- Distribution Schema
- Support for XA interface

The version of SQL/CLI that RDA3 currently supports is that of ISO 9075-3:1995 which is commonly known as SQL2. The next version of RDA, informally known as RDA4, will be based on the CLI of SQL3. A working draft of RDA4 (dated, 98-03-20) already exists.

Requirements for incorporating security facilities in RDA have been duly identified. A security amendment called Secure RDA (FPDAM, 12 March 1998) has been balloted and approved. For more details on RDA Facilities for Security see Annex D.

The work on a distribution schema for a distribution controller is still at a preliminary stage. The distribution schema plays an analogous role to the SQL Information Schema in that it provides an implementation independent mechanism for determining the content of a database.

There is no independent concept of an RDA Transaction, but RDA is designed to work with external encompassing transaction managers if required. RDA can be enhanced by defining support for XA Transaction Managers, other transaction management schemes, improving RDA performance in high transaction environments, and providing facilities for transmitting heuristics. A draft specification exists showing how RDA can optionally be used in conjunction with encompassing transaction managers.

It is not quite clear at this time how these enhancements will be progressed, whether as Amendments one at a time or as integral parts of the next version of RDA.

11. Maintenance and Promotion

The Library of Congress is designated as Maintenance Agency and Registration Authority for ANSI/NISO Standard Z39.50 and ISO 23950.

Formal responsibilities of the Maintenance Agency include development and processing of, and provision of access to:

- Defect Reports
- Amendments
- Clarifications
- Profiles

As Registration Authority, its formal responsibilities include maintenance of :

- Objects and External Definitions

And in addition, its informal responsibilities include development and processing of, and provision of access to :

- Register of Implementors
- Implementor Agreements
- ZIG Commentaries
- Hosts Available for Testing
- Information Regarding Availability of Z39.50 Software
- Miscellaneous information.

RDA has no equivalent organization providing maintenance agency functions. The ISO/IEC JTC1/SC32/WG5 is responsible for maintaining the RDA standard. RDA being an ISO/IEC standard its amendments, defect reports, clarifications are handled by SC32/WG5 through the established ISO procedures for handling such matters. The maintenance process for the latest version of RDA (ie, RDA3) is far less complicated than that for the Z39.50 as RDA3 has no need to define profiles, no need to define OIDs and register External Definitions.

12. Conclusions

Z39.50 was originally designed to search and retrieve bibliographic records. As its purpose was limited so was its functionality. With the passage of time the temptation to add more functions and apply it to other applications proved irresistible. In the process the protocol specification got greatly extended giving rise to the need for definition of profiles, registration of externally defined information objects, adding new features and options, etc. However, the limitations of the original design could not be easily overcome, and the protocol specification continues to grow in size and complexity with each new version. The RDA specification went through almost the reverse process.

The initiative for Z39.50 came from user organizations like large academic and public libraries, and not from mainstream computer companies and database vendors. User organizations continue to exercise control over the direction of its enhancement, maintenance, and promotion. This state of affaires combined with a benign neglect by the major vendors in the field of database technology may well have been an advantage for Z39.50. For RDA1 and RDA2, spanning nearly a decade of development, the situation was quite different. Major database vendors rather than user organizations strongly affected the design decisions and the eventual fate of the RDA2 standard.

Database vendors may soon realize that it is to their commercial advantage to provide more than one external interface in their products. Major vendors are offering SQL front-ends to their non-relational DBMS products. Other information provider organizations may allow their database servers to serve both Z39.50 clients and RDA clients.

There is no perfect all-purpose protocol suitable for every kind of information retrieval or remote database access application, and technical merit alone seldom decides the fate of a standard. It is probably safe to say that the world of databases and repositories of useful information will never be uniform and will never conform to any single model. In such a mixed environment there will be place for more than one information retrieval or remote database access protocol. There is no very strong reason why both Z39.50 and RDA can not coexist and flourish.

One must also wonder whether it is not in the best long term interest of user organizations to rely on mainstream off-the-shelf technology and vendor provided support rather than get involved in custom designed, user funded and supported technology. Commitment to using Z39.50 is bound to involve user organizations heavily in the types of responsibilities and associated expenses for which they are admittedly not as well equipped as database vendors.

User organizations may also wish to take a somewhat longer term view of their current and possible future applications. As old technology gives place to new, systems will move towards the use of database technology, from centralized to distributed databases, from read only to update and transaction processing, from dealing with global "attributes" and whole "records" to dealing with multi-media "content" as well. There will be need for ad hoc queries as well as for retrieving files of "records". And there will be a greater flow of electronic information among various sectors of economy. For all such applications RDA provides in my view a far more attractive and potentially less costly path to future growth and some insurance against technological obsolescence.

13. References

1. CONSULTING AND AUDIT CANADA, Statement of Work, Contract No. : 510-8160, Project No. : 510-2180, Vendor No.: 01Z5G6, Date : May 20, 1997.

2. ANSI/NISO Z39.50-1992 (Revision of ANSI/NISO Z39.50-1988), Information Retrieval Application Service Definition and Protocol Specification for Open Systems Interconnection.

3. ANSI/NISO Z39.50-1995 (Revision of ANSI/NISO Z39.50-1992), Information Retrieval Application Service Definition and Protocol Specification.

4. Z39.50 Maintenance Agency Procedures, January 6, 1999
   Note : This document is the principal source of information about all aspects of the Z39.50 standard and its implementations. The maintenance Agency for Z39.50 is the U.S. Library of Congress.

5. RFC 1729 [December 1994] : Using the Z39.50 Information Retrieval Protocol in the Internet Environment.

6. Clifford A. Lynch : The Z39.50 Information Retrieval Standard Part 1: A Strategic View of Its Past, Present and Future (D-Lib Magazine, April 1997)

7. Robert M. Colomb and Sonia M. Finnigan : Z39.50/SQL+ Project (Source : http://www.dstc.edu.au/)

8. Sonia Finnigan, Linda Bird, Robert Bull : Z+SQL ñ Issue 2, 8 February, 1999

9. ISO/IEC 9579-1 : 1993, Remote Database Access - Part 1 : Generic Model, Service and Protocol

10. ISO/IEC 9579-2 : 1997, Remote Database Access - Part 2 : SQL Specialization.

11. ISO/IEC 9579 : 1999, Remote Database Access for SQL.

Annex A : Z39.50 Information Retrieval Protocol [5]

The Z39.50 standard is intended for systems supporting information retrieval services, for organizations such as information services, universities, libraries, and union catalogue centers. It addresses connection-oriented, program-to-program communication. It does not address the interchange of information with terminals or via other physical media.

The client may initiate requests on behalf of a user; the protocol addresses communication between corresponding information retrieval applications, the client and server (which may reside on different computers); it does not address interaction between the client and user.

Z39.50-1992 provides the following basic capabilities, all of which are supported in Z39.50-1995 as well. The client may send a search, indicating one or more databases, and including a query as well as parameters which determine whether records identified by the search should be returned as part of the identified and possibly some or all of the records. The client may then retrieve selected records. The client assumes that records selected by the search form a “result set” (an ordered set, order determined by the server), and records may be referenced by position within the set.

Facilities of Information Retrieval Service :

Z39.50 Version 3 provides eleven facilities :

NOTE : Those marked with (**) are new, that is, only in Version 3.

Initialization Facility-- *Init Service* **:** allows the origin to establish a Z-association. The origin proposes values for initialization parameters ie, version, authentication, facility capabilities. In the *Init* response, the target responds with values for the initialization parameters; those values, which may differ from the origin-proposed values, are in effect for the Z-association.

Search Facility -- *Search Service:* enables an origin to query databases at a target system, and to receive information about the results of the query. The *search* request allows the origin to request that the target apply a query to a specified set of databases at the target, to identify records with the properties indicated by the query. The target creates a result set, which represents the set of records identified by the query, and the target maintains the result set for subsequent retrieval requests. Depending on the parameters of the search, one or more records of the result set may be immediately retrieved as part of the *search* response. The result set is an ordered set; a record identified by an entry in the result set is referenced by the position of the entry within the result set.

Retrieval Facility -- consists of two services :

Present Service: allows the origin to request response records corresponding to database records represented by a specified result set. Database records are referenced by relative position within the result set. The origin specifies a range/several ranges and may follow with subsequent requests specifying different ranges.
Segmentation Service: if the records requested by a *present* request will not fit in a single segment, and if segmentation is in effect, the target returns multiple segments, each of which contains a portion of the records.

Result-set-delete Facility -- *Delete Service:* enables an origin to request that the target delete specified result sets, or all result sets. The target responds by reporting information pertaining to the result of the operation.

Access Control Facility -- *Access Control Service:* allows a target to challenge an origin. The challenge might pertain to a specific operation or to the Z-association. The *access-control* request/response mechanism can be used to support access control challenges or authentication, including password challenges, public key cryptosystems, and algorithmic authentication.

Accounting/Resource Control Facility -- consists of three services :
Resource-control Service: permits the target to send a *resource-control* request, which might include a *resource report.* The report might notify the origin that either actual or predicted resource consumption will exceed agreed upon limits (or limits built into the target), and request the origin's consent to continue an operation, via the *resource-control* response. The target might,

for example, inform the origin about the current status of a result set being generated on the target during a *search* operation, and indicate information about the progress of the operation.
Trigger-resource-control Service: permits the origin to request that the target initiate the *resource-control* service, or cancel the operation pertaining to a completed operation or to the Z-association.
Resource-report Service: permits the origin to request that the target send a *resource-report*

****Sort Facility** -- *Sort Service:* allows an origin to request that the target sort a result set (or merge multiple result sets and then sort). The origin specifies a sequence of sort elements. The result set is to be ordered according to the specified sequence, and subsequent positional requests against the result set will be construed by the target to apply to the result set as so ordered.

****Browse Facility** -- *Scan Service:* used to scan an ordered term list (subject terms, names, titles, etc.). The ordering of the term list is target defined. The origin specifies a term-list to scan and a starting term (implicitly, by specifying an attribute/term combination and a database-id), the size of the scanning steps, and the desired number of entries and position of the starting term in the response.

****Explain Facility** -- This facility does not include any services, but uses the services of the *search* and *retrieval* facilities to allow the origin to obtain details of the target implementation, including general features (description, contact information, hours of operation, restrictions, usage cost, etc.), databases available for searching, indexes, attribute sets, attribute details, schemas, record syntaxes, element specifications, sort capabilities and extended services supported. The target maintains this information in a special database, *IR-Explain* (with a predefined record syntax), that the origin may access as any other database supported by the target.

IR-Explain is made up of 15 categories, each of which provides different information about the server. The *TargetInfo,* for example, supplies general information about the server, and the *DatabaseInfo* category supplies database-specific information. Each category is implemented independently and there is, therefore, no need to support all of the categories. For interoperability, the *CategoryList* category provides a convenient mechanism for a client to determine what categories are supported by a server.

****Extended Services Facility** --** *Extended Services Service:* allows an origin to create, modify, or delete a task package at the target, which may persist after the Z-association has been terminated. The target maintains task packages in a special database, *IR-Extend-I,* which has a predefined record syntax. An extended service is a task type, related to information retrieval, but not defined as a Z39.50 service. Execution of a task by the target is outside the scope of Z39.50.

The Extended Services defined by this standard are :

- Save a result set for later use
- Save a Query for later use
- Define a periodic search schedule
- Order an item
- Update a database
- Create an export specification
- Invoke a previously created export specification

The origin sends an *ES* Request to the target requesting execution of a task. The request includes parameters which the target uses to construct the task package. The target checks the request for validity, for consistency with the user's access privileges, and possibly for other target-dependent limitations. The target sends an *ES* response indicating that the request was accepted or supplying an indication of the reason the request was rejected. The origin may access the extended service database, *IR-Extend-I*, as any other database supported by the target.

Termination Facility -- *Close Service:* allows either an origin or target to abruptly terminate all active operations and to initiate termination of the Z-association.

Query Formulation :

The Z39.50 standard at present specifies six query types:

(1) *Type 0* - designated "private", allowing two systems to use a private,

mutually agreed upon query format.

(2) *Type 1* - queries are expressed by individual search terms, each with a set of attributes. Terms may be combined/linked by boolean operators. Terms and operators are expressed in Reverse Polish Notation.

(3) *Type 2* - specified by ISO 8777 - Commands for lnteractive Text Searching.

4) *Type 100* - specified by ANSI Z39.58- Common Command Language for Online Interactive Information Retrieval.

5) *Type 101* - extension of type-1 query for proximity searching.

(6) *Type 102* - ranked list query (undefined at present).

Z39.50 fully specifies and mandates support of the Type-1 query.

Attribute Sets :

The attributes associated with a search term belong to a particular ***Attribute Set***, whose definition is registered, that is, assigned a unique and globally recognized attribute-set-id, an Object Identifier, which is included in the query. Attribute Sets provide a virtual database representation of their domain. For example, the ***Bib-1*** attribute set was developed for the bibliographic community, to provide a common, abstract model by which to view differing library systems for the purpose of searching and retrieving information in standard and mutually understandable terms. Attribute sets supported at present are:

Bib-l	- for bibliographic use
Exp-1	- for use with an Explain database
Ext- I	- for use with an Extended Service database
CCL-1	- Common Command Language
GILS	- Government Information Locator Service
STAS	- Scientific and Technical Attribute Set

Response Record :

The protocol distinguishes two types of response records that may be returned by the server: **database** and **diagnostic** records. Database records may be returned in several formats. Like attribute sets, each format's definition is also registered, that is, assigned an Object Identifier, a unique and globally recognized attribute-set-id.

(1) bibliographic syntaxes not described via ASN.1 (transfer-syntax ISO 2709 may be used) various MARC formats, including USMARC, UNIMARC, UKMARC, and CANMARC.

(2) syntaxes which are described via ASN.1 (transfer-syntax ISO 8825 may be used):

Explain	- Server Information syntax
ESTask Package	- Extended Services record syntax
SUTRS	- Simple Unstructured Text Record Syntax (text only)
OPAC	- Online Public Access Catalogue
Summary	- Bibliographic Summary syntax
GRS 1	- Generic Record Syntax, to return records with structure

Diagnostic records are similarly accompanied by object identifiers which identify their format;

(1) error formats	- bib-I, diag-1
(2) resource report	- resource-1, resource-2
(3) access control	- prompt-i, des-i, krb-l

Optional capabilities include:

• The client may specify an *element set* indicating data elements to retrieve in cases where the client does not wish to receive complete database records. For example, the client might specify “If 5 or less records are identified, transmit ’full’ records; if more than 5 records are found, transmit ’brief’ records”.
• The client may indicate a *preferred syntax* for response records, for example, USMARC.
• The client may *name* a result set for subsequent reference.

• The client may *delete* a named result set.
• The server may impose *access control* restrictions on the client, by demanding authentication before processing a request.
• The server may provide *resource control* by sending an unsolicited or solicited status report; the server may suspend processing and allow the client to indicate whether to continue.

Z39.50-1995 New Features [3]

Provided below is a summary of the enhancements in Z39.50-1995. The designations “version 2” and “version 3” refer to protocol version; “Z39.50-1992” and “Z39.50-1995” refer to the respective standards. Thus where a particular feature is described as “new in Z39.50-1995”, that generally means it applies in either protocol version. An example is **Scan**: an implementor may add the Scan service to an existing implementation of Z39.50-1992 without incorporating any other new features. The enhancements described below fall into four categories: search, retrieval, new services and facilities, and miscellaneous enhancements.

Search

Attributes
There are a number of enhancements pertaining to attributes and attribute sets. In version 3, attributes may be combined from different attribute sets, within a single query (even for a single search term). This presents two advantages: First, it is useful when searching multiple databases. (Although version 2 supports multiple-database searches, all attributes within a query must belong to a single attribute set, which inhibits the ability to search multiple databases, unless those databases are similar.) Second, new attribute sets may now be defined with less replication. Version 3 provides two further enhancements allowing flexibility in the definition of attribute sets. First, new data types for attribute values are defined an attribute set definition may now list alternative sets of evaluation rules (for example, whether the server is allowed to substitute an attribute that it thinks is more appropriate), and the query may select one of the alternatives. The enhanced bib-1 attribute set definition exploits this new feature. The bib-1 definition in Z39.50-1995 also includes many new attributes (as well as all of the attributes in Z39.50-1992).

Extended Result Set Model

The basic model of a result set is developed in Z39.50-1992; the 1995 version describes an “extended result set model”, which supports extended proximity searching. The extended model also supports a new version 3 search function, *restriction*, which is (in effect) an operation on a result set. It permits selection of records from a result set, based on specified attributes.

Search Term

The search term for a query may take on a variety of data types in version 3. (In version 2 a search terms is binary and thus essentially has no data type, so the type is often described by a structure attribute.) This enhancement will simplify queries (as well as attribute set definitions) by reducing the need for structure attributes.

Intermediate Results

In Z39.50-1995 the server may provide information per query *component* (i.e. per sub-query, per database), as part of the Search response (version 3 only), or as part of resource-control when the server reports on the progress of the search. The server may also create and provide access to a result set for individual query components.

Retrieval

Segmentation

In version 2, a retrieval response is limited to a single message; the server attempts to fit the requested records into the message, and if it cannot, it simply fits as may as it can. The client might want to retrieve, for example, ten thousand records, knowing it cannot retrieve them in a single message.

Typically the client will request all ten thousand records, wait for the response, determine how many records are retrieved, and then send another request for the remaining records. This works well in many environments but is unacceptably slow for high-speed sending each set of records, which introduces a delay; the delay may be negligible for conventional networks, but is intolerable for high-speed networks.

In version 3 a server may respond to a retrieval request with multiple consecutive response messages without intervening requests. A more serious segmentation problem occurs when a single record is too large to fit in a message. Version 3 thus introduces a second level of segmentation: an individual record may span response messages. A client or server may choose to support either level of segmentation, or no segmentation (in which case version 2 rules apply).

Retrieval Tools

The ZIG has worked intensively over two years to develop an extensive model and suite of tools for a wide range of retrieval functions to support various retrieval applications, in particular, document retrieval. The model is detailed in Appendix RET. Several new object classes are designated in Z39.50-1995 (schemas, tagSets, variants) and specific objects from these and other classes are defined. Appendix RET provides detailed semantics for these objects and describes how they are used together to provide a variety of document retrieval capabilities. Following are a few examples:

- A single database record might include a number of documents. The client may discover and retrieve a specific document, rather than the entire database record.
- The client may retrieve a specific portion of a document, logical or physical, for example, specific pages, a specific chapter, a specific caption, all captions, or all images. The client might retrieve just *headings,* for example, all chapter or section headings.
- A document might be available in a wide variety of formats (e.g. postScript, SGML), languages, presentation parameter (e.g. line length, lines per page, columns), and other variants. The client may discover what as information associated with a particular variant form: for example the cost to retrieve the document according to a specific variant, or its size. Finally, the client may then retrieve the document (or specific portion) according to the desired variant.
- Associated with a document, for a given search, may be *hits*: pointers to terms (within the document) relevant to the search. The client might retrieve hits along with a document to quickly locate the satisfying portions. Or the client might retrieve only the hits (ranked in order of importance), and subsequently retrieve only the indicated satisfying portions.

New Services and Facilities [3]

Scan and Sort

Scan and Sort are new services in Z39.50-1995. These are used respectively to scan terms in a list or index, and to sort a result set. Scan is currently the only service in the Z39.50 **Browse** facility, but it is intended that various other browse capabilities will be added in future versions.

Extended Services
Extended Services is a new facility in Z39.50-1995. It includes a new Z39.50 service, the *Extended Services service*, used to initiate a specific extended service task, which is executed outside of the Z39.50 session and whose progress may be monitored using Z39.50 services. Specific extended services include:
- save a result set,
- set a periodic query schedule,
- export a document,
- order a document, and
- update a database.

Explain
The new Explain facility allows a client to retrieve details of the server implementation: general features (description, contact information, hours of operation, restrictions, usage cost, etc.) databases available for searching, indexes, attribute sets, attribute details, schemas, record syntaxes, sort capabilities and extended services. The server maintains Explain information in a special database that may be accessed by the client using the Z39.50 search and retrieval facilities. The format of the Explain information is detailed in the standard.

Some Explain information is transparent to the client, intended for direct display to the client-user, and is so designated (e.g. “general features”). Some Explain information is intended to be shared by client and user. For example, the client may retrieve a list of searchable databases; for each database in the list the client might display an *informal* name, an icon, and a brief description. Meanwhile the client would retain the *actual* database name to be used in a protocol message, which probably would not be displayed.

Some Explain information may be completely transparent to the user. For example, the client may retrieve information about attributes supported for a database and use that information when formulating a query (when converting a user-supplied query to a Z39.50 type-1 query).

Miscellaneous Enhancements

Termination and Re-initialization
Version 3 includes a more flexible approach to termination of a Z39.50 session, to allow, in effect, re-initialization without taking down the network connection.
Concurrent Operations
Multiple concurrent operations are allowed in version 3. In version 2, operations are strictly serial.
Diagnostics
Most Z39.50 services include diagnostic capability. Specific format defined within the standard. In version 3, diagnostic formats may be externally defined and registered. One such (new) format is defined, along with a comprehensive set of diagnostics.
Access Control Formats
Z39.50-1992 provides access control, but does not define any access control formats. Z39.50-1995 defines formats for encryption and authentication, and a format allowing the server to prompt the client for arbitrary information.
Character Set Support
A new data type, “International String”, has been introduced for character strings. Its definition allows greater flexibility for a client and server to agree to the use of a particular language and one or more character sets during a session.
Units
New data types are introduced for support of units. These definitions allow standard representations to be used to represent unit type and unit. For exam-ple, unit type might be “mass”, and unit, “kilogram”.

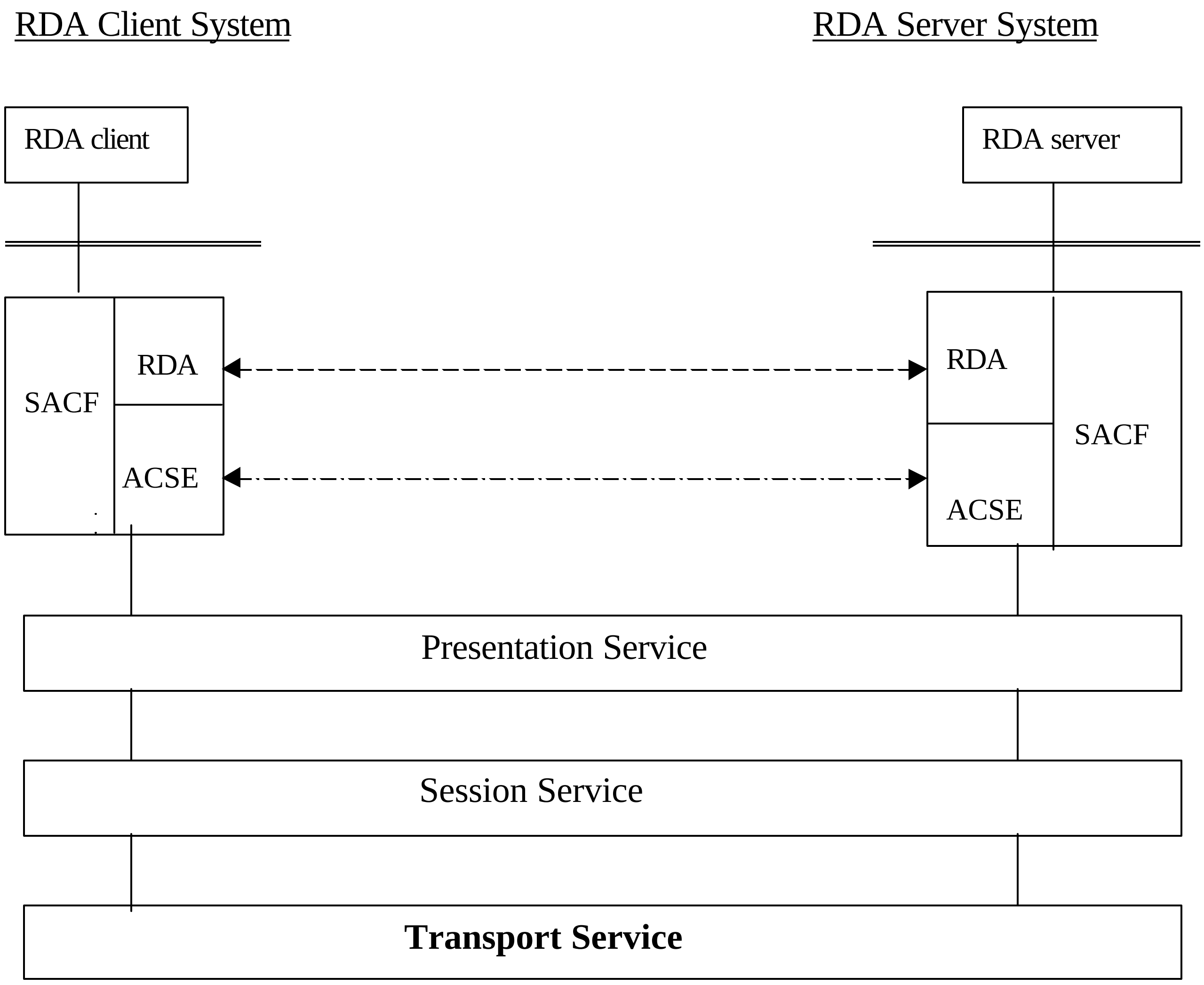
Extensibility and Negotiation
Version 3 provides a powerful extensibility feature. Each protocol message includes a field designated for information whose format is to be defined externally. These externally defined formats will be registered and maintained by the Z39.50 Maintenance Agency, as provisional extensions to the standard, for experimental use and possible consolidation into a subsequent version.

In Z39.50-1995 the concept of a “negotiation record” is introduced. The client may include a negotiation record within the initialization message to propose The negotiation record is an application of the new extensibility feature. Negotiation records will be defined externally and maintained by the Z39.50 Maintenance Agency.

Annex B : RDA

B1 : RDA2 (ISO/IEC 9579-1, -2, -3)

The following diagram indicates the upper layer structure of the RDA2 Service. The RDA protocol operates within the A-Association controlled by the ACSE (Association Control Service Element). RDA must also rely on the services provided by the Presentatation layer and all the layers below. In view of its upper layer structure RDA2 can not be mapped directly on the transport services provided by TCP/IP.



Annex B2 : RDA3 (ISO/IEC 9579:1999) [11]

1 Scope

This International Standard, Remote Database Access for SQL (RDA), defines a model for the remote interaction of an SQL-client and one or more SQL-servers through communication media, and defines the encoding of messages, the semantics of messages and associated facilities for mediating the interaction between one SQL-client and one SQL-server.

This International Standard also defines a mapping of the RDA Protocol to the specific communication infrastructures TCP/IP and Transport Layer Security (TLS).

This International Standard relies upon the facilities provided by ISO/IEC 9075 (SQL) and ISO/IEC 9075-3 (SQL/CLI).

Normative annexes provide:

- a Conformance Proforma,

- an optional language independent Application Programming Interface defined in the notational conventions of ISO/IEC 9075-3 (SQLICLI) for invoking RDA Operations,

- an optional mapping of ISO/IEC 9075-3 (SQL/CLI) functions to RDA Operations,

- definitions of optional SQL-servers, the RDA Location Server and the RDA Support Server, to facilitate interoperation and data distribution in a heterogeneous environment.

Informative annexes provide:

- an ASN.1 specification for the RDA Protocol,

- an ASN.1 specification for the encoding of multiple rows.

This International Standard does not constrain:

- conforming RDA-client environments to be implemented using any particular processor decomposition,
- conforming RDA-server environments to be implemented using any particular processor decomposition.

This International Standard does not define:

- algorithms for query decomposition or for the combining of results in a distributed database environment,
- recovery mechanisms in the event that transaction co-ordination fails.

5.1 Model

ISO/IEC *9075* (SQL) defines an SQL-environment, SQL-client and SQL-server. This International Standard (RDA) defines how an SQL-client and SQL-servers interact when supported by an underlying communication system. Concrete definitions for some aspects of the SQL model and some SQL concepts are given to the extent that is required for remote, heterogeneous, interoperation of an SQL-client and an SQL-server using the facilities defined by this International Standard, RDA.

The model adds the following key concepts**:**

RDA-client,
RDA-server,
Transport Provider, and
Transport Connection.
Subsequent subclauses of this clause define each of these concepts.

Figure l provides a representation of the interrelationship of these components and related SQL terms.

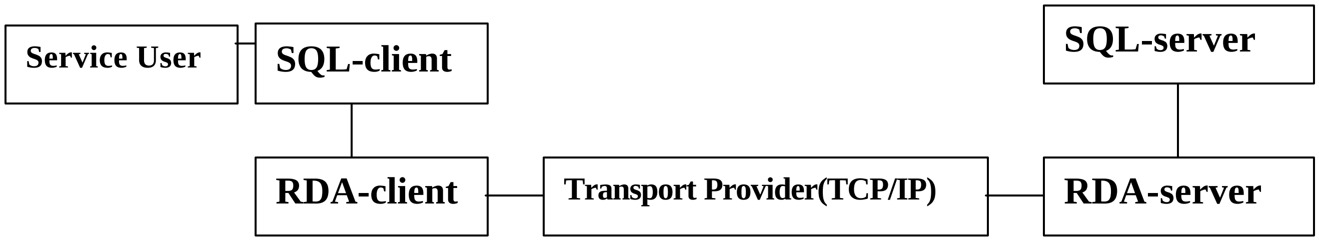


Figure 1 - RDA3 model of SQL-environment

The model adopted by this International Standard is that an RDA-client environment, which includes a Service User and an SQL-client, uses one or more Transport Providers to connect to RDA-server environments, each of which has one or more SQL-servers associated with it thereby establishing SQL-sessions between the SQL-client and the SQL-servers. Each SQL-session is associated with an SQL-connection. These are in turn supported by Transport Connections within Transport Providers that are established and terminated as required for communicating through the SQL-connection.

More than one Transport Provider may be used concurrently to support concurrent SQL-sessions. A particular SQL-server may be in more than one RDA-server environment so the particular Transport Provider used to support a session between an SQL-client and a particular SQL-server may differ between SQL-sessions.

Annex C : Draft Proposal for SQL Extension to Z39.50

by
Sonya M. Finnigan, Robert M. Colomb
Distributed Database Unit, CRC for Distributed Systems Technology Centre
School of Information Technology The University of Queensland, Qld 4072, Australia
Email: s.finnigan@dstc. edu.au

NOTE : The following is a short excerpt from the above paper.

Introduction

Z39.5OISQL+ proposes SQL extensions to the ANSI/NISO Z39.50-1995 for inclusion into Version 4. The proposal extends Z39.50 to support SQL querying and the SQL export record syntax. Currently, the proof-of-concept Z39.50/SQL+ prototype, called "Zinc", is being extended to support the Australian museum community. A test-bed to support the GILS profile is also under consideration.

The SQL extensions are as follows:
- A new query type, *type-SQL* supporting SQL3
- A new attribute type, *fieldName*
- Additions to the Explain Syntax

In addition, two supporting record syntaxes are proposed:
- *SQL-RS,* supporting SQL3 datatypes
- *ERR-RS*

The above extensions are described in detail in the (following) description of Z39.50/SQL+.

General Discussion

Z39.50 recognizes that information retrieval consists of two primary components -- selection of information based upon some criteria and retrieval of that information, and it provides a common language for both activities. Z39.50 standardizes the manner in which a client and a server communicate and interoperate even when there may be differences between computer systems, search engines, and databases.

Z39.50/SQL+ can be seen as an extension of the existing *Z39.50-1995* (Version3) protocol, uniting the advantages of SQL's query language and export syntax with the information retrieval services of Z39.50. The SQL extension provides a standardized way of specifying complex structured queries which otherwise are not possible with the existing Z39.50 query types. In addition, result sets (which may contain complex data types) may be returned in a generic record syntax without having to tag each individual field of each record and without being tied to a pre-defined schema. The *Z39.5OISQL+* standard remains platform and database independent. Indeed, the supporting database may be an RDBMS, OODBMS, O-RDBMS or even a text database for that matter. As is now the case with Z39.50, the onus is on the server to map the Z39.5O PDUs to the applicable database calls.

Z39.50/SQL+ provides the Z39.50 client with the full flexibility and query power of SQL. *Z39.5OISQL+* clients are able to specify complex queries, either by using SQL or one of its derivatives, such as Query-by-Example (QBE). Queries can be formulated on single (virtual) tables, as in now the case, or multiple tables supporting cartesian products, unions, intersections, joins on matching columns, and projections on given columns. Queries can also be formulated using powerful constructs for expressing conditions, performing aggregate and comparison operations. partitioning tables into groups and much more.

Z39.50/SQL+, like Open Database Connectivity (ODBC) and Remote Database Access (RDA) Protocol Part 2 - SQL Specialization, uses SQL as its standard way of expressing complex queries on structured data. However, it should be noted that the functionality of z39.50/SQL+ is conceptually different from that of ODBC and RDA. Both of these simply convey SQL statements (both retrievals and updates) and related control operations (including concurrency control and synchronization) to the database management system. In contrast, Z39.50/SQL+ is concerned with interoperable information retrieval, and facilitates the management of state within and across communication sessions, a standard means of dynamically accessing the existing catalogues, the option of using a standardized schemas for interoperability, as well as extra facilities such as scan and extended services.

Z39.5OISQL+ provides additional querying and retrieval power to existing Z39.50 implementors as well as expands the usage capabilities of Z39.50 away from the library-type communities, which are mostly text based, to a much broader community employing a wider class of database as part of their information management resource base.

A particular example of a new class of service that *Z39.50/SQL+* facilitates is web-based data mining on a data warehouse [Matheus et al. 1993]. This technology uses algorithms related to SQL aggregation to identify patterns in large data sets. It is very computation-intensive and relies on successive data reductions on the server to extract a relatively small pattern. A group of major retailers could make their data warehouses available as (probably expensive) commercial services, which would cater to market researchers, who might wish to broadcast a data mining query. Furthermore, aggregation could support valuable content analysis queries on text databases [Weber et al. 1990], such as identification of the thousand most mentioned concepts in a newspaper archive, and trends in the pattern of number of mentions of each year by year over ten years.

What ís New :
Z39.5OISQL+ introduces a new query type, a *type-SQL* query - conforming to the <query expression> of the SQL3 Standard [ISOIIEC *9075-2.1996].* The SQL <query expression> is highly structured, allowing search terms and attributes to be specified within the query. The Z39.50 search APDU requests the results of the SQL query be stored in the <tableName> (as defined in the SQL standard) and specified as per the
result set name option. The actual implementation mechanism of this request would be the responsibility of the server, as is now the case with the existing *type-1* query. (An example of a possible mapping to a RDBMS is listed in Appendix III.) The server could then describe its implementation mechanism and what version of SQL it supports within the Explain facility.

Z39.50/SQL+ introduces a new attribute *type*, *fieldName* within the Attribute Set Architecture. The *fieldName* attribute is based on the "classical" model where we presume a close relationship of access points and retrievable fields. Each element listed in the *fleldName* attribute is unique within an Attribute Set and has a name (semantically equivalent to a <field Name> as defined in the SQL standard), a numeric value, and a direct mapping to an existing schema element.

Element Sets follow the same approach - being a direct subset of *fleldName* elements. If supported, a specified element set is retrieved without individually specifying the exact elements within the SQL query.

However, it should be noted, that the *fieldName* attribute need not be supported as many server databases already store their schema metadata within their system catalogues. These catalogues could be accessed via the Explain facility for dynamic building of client interfaces.

Z39.5OISQL+, therefore, offers the flexibility of either using the *fieldName* attribute within the *type-SQL* query expression or using dynamically generated field names directly from a database - under the same client interface.

Like Z39.50, Z39.50/SQL+ still distinguishes two types of response records that may occur from the server: database and diagnostic records. It introduces a new record syntax, *SQL-RS,* by which database records may be returned, and similarly an additional error format, *SQL-ERR. SQL-RS* conforms to an extended version of the export record syntax of the Remote Database Access (RDA) Protocol Part 2 - SQL Specialization [ISOIIEC 9579.2, Amend 1 CD *'95]* with support for SQL3 datatypes. SQL-RS is a generic record syntax for returning structured records. The record syntax eliminates the need to tag each individual field for each record and offers the flexibility of not being tied to a predefined schema.

Minor extensions to the Explain record syntax include SQL version, database version and catalogue table name parameters.

We are currently investigating the worth of, and issues involved in, tokenizing the SQL query so as to be language independent.

Query Formulation

Z39.50/SQL+ introduces a new query type, *type-SQL.* The *type-SQL* query consists of the <query expression> of the SQL3 Standard [ISO/IEC

9075.2:1, CD-1996]. The SQL <query expression> is highly structured allowing search terms and attributes to be specified within the query. The Z39.50 search APDU would request the results of the SQL query be stored in the <tableName> (as defined in the SQL standard) and specified as per the result set name option.

Type-SQL queries can be formulated on single (virtual) tables, as in now the case, or multiple tables sup-porting cartesian products, unions, intersections, joins on matching columns, and projections on given columns. Queries can also be formulated using powerful constructs for expressing conditions, performing aggregate and comparison operations, partitioning tables into groups and much more. Many of these querying capabilities are not possible within the *type-1* boolean query.

It could be argued that the "private" *type- 0* query would suffice, however this would greatly reduce the interoperability of the extension, restricting client/server communication to only those which allow specific conversions.

Annex D : RDA Security

(From Working Draft of Fourth Edition (20 March 1998) of
ISO/IEC 9579:199x)

5.8 RDA Facilities for Security

51.1 RDA Security Services

The requirements for security services are discussed in Annex A.

These services are:

RDA Service User Authentication: corroboration of the identity of the Service User.

RDA-client Authentication: corroboration of the identity of the RDA-client.

RDA-server Authentication: corroboration of the identity of the RDA-server.

RDA Outgoing Access Control: access control enforced by the RDA-client Environment.

RDA Firewall Access Control: access control enforced by the RDA relay system **.**

RDA Incoming Access Control: access control enforced by the RDA-server Environment.

SQL Access Control: access control enforced by the SQL-server.

RDA Transfer Integrity: protection of communication against unauthorized modification.

RDA Transfer Confidentiality: protection of communication against unauthorized disclosure.

RDA Request non-repudiation: protection against the RDA-client denying having requested a specific RDA Operation.

RDA Response Non-repudiation: protection against the RDA-server denying having responded to a specific request for an RDA Operation.

The following subelauses describe how facilities defined or identified by this International Standard may be used to provide the security services. Profiles of combinations of services to achieve defined levels of security are defined in Annex A.

5.8.2 Use of Transport Provider security facilities

The Transport Provider may have facilities for providing RDA Transfer Integrity, RDA Transfer Confidentiality, RDA-client Authentication and RDA-server Authentication.

A Transport Mapping is defined in 10.2 that can be used to provide RDA Transfer Integrity, RDA Transfer Confidentiality, RDA-client Authentication and RDA-server Authentication.

In cases where the RDA Client represents a single Service User then RDA Client Authentication may be used to support RDA Service User Authentication as described in *5.8.3*.

5.8.3 Use of Authentication in RDA Connect

The RDAConnect operation fields AuthenticationType and Authentication can be used to authenticate the name of the RDA Service User as identified by the UserName field.

Four types of RDA Service User Authentication, identified by the AuthenticationType, arc supported:

- *password*. the User Name is authenticated using a password carried in the Authentication field.

- *transfer*. the UserName is matched against the RDA-client identity authenticated by RDAclient Authentication (for example using a local equivalence table or mapping algorithm). This mechanism requires that the 'WA-client represents a single Service User.

NOTE 18 - The Service User may, however, use several User Names to represent different privileges and may be represented by several RDA-clients.

- *attributeCertificate*, the User Name is related to the identity of an RDA-client by an Attribute Certificate (as defined in ISO/IEC 9598-8 X.509 1997) carried in the Authentication field. This mechanism requires that the RDA-client represents a single Service User.

NOTE 19 - The Service User may, however, use several User Names to represent different privileges and may be represented by several RDA-clients. Also, if Service Users share common privileges, and do not need to be separately identified for other reasons (for example, accountability), then they can be treated as the same Service User for the purposes of authentication, sharing a common UserName authenticated through the same RDA-client

- *other*, the Authentication field is used to carry authentication information to support an externally defined mechanism.

The AuthenticationType may also be set to *none* to indicate that no support is provided for RDA User Authentication.

An Attribute is provided to permit an RDA-client to request a particular type of RDA Service User Authentication:

AUTHENTICATION TYPE: a value defining the type of RDA Service User Authentication requested by the RDA-client Environment.

Use of MessageAuthentication in RDAMessage

Each RDAMessage protocol element has a field MessageAuthentication that provides a means of communicating information required for RDA Request Non-repudiation and RDA Response Non-repudiation.

Three levels of RDA Request Non-repudiation and RDA Response Non-repudiation are supported:

- *none*, no support is provided.

- *originatorSigned*, RDA Non-repudiation is supported by a timestamp and signature produced by the originator of the message.

- *ttpSigned*, RDA Non-repudiation is supported by a timestamp and signature produced by the originator of the message, together with a timestamp and signature produced by a trusted third party (TTP).

Attributes are provided to permit an RDA-client and an RDA-server to negotiate the RDA Request Non-repudiation level and the RDA Response Non-repudiation level, and for a Service User to request a particular level of RDA Request Non-repudiation and RDA Response Non-repudiation.

REQUEST NON-REPUDIATION PROVIDED: a value defining the level of RDA Request Non-repudiation provided by the RDA-client Environment.

REQUEST NON-REPUDIATION REQUIRED: a value defining the level of RDA Request Non-repudiation required by the RDA-server Environment.

RESPONSE NON-REPUDIATION SUPPORTED: a value defining the level of RDA Response Non-repudiation supported by the RDA-client Environment.

RESPONSE NON-REPUDIATION REQUIRED: a value defining the level of RDA Response Non-repudiation required by the RDA-client Environment.