

Protocol Support for Commercial Access to Complex Database Applications

Robert M. Colomb

Distributed Systems Technology Centre Pty Ltd
School of Information Technology
The University of Queensland
Queensland 4072 Australia
colomb@cs.uq.edu.au

Sonya Finnigan

Distributed Systems Technology Centre Pty Ltd
sonya@dstc.edu.au

Primary categories: Protocol Development, Database/Web Gateways

Also relevant to: Charging and E-commerce, Tools and Browser Enhancements, Information Retrieval, Resource Discovery, Human-Computer Interaction, Digital Libraries

Abstract

This paper examines the problems involved in developing applications for the Web which require complex and stateful dialogues between a user and a service. The application-layer Z39.50 protocol is shown to satisfy most of the requirements for text-style databases. We describe proposed modifications to the protocol to support access to structured (SQL) databases, and a prototype implementation of the client and server software needed to support the modified protocol.

Introduction¹

Now that people are getting over the novelty of being on the Web, there is room to give attention to the next steps. This paper addresses two potential developments. First, many Web sites are front ends to databases, but most such sites permit only a simple request/response interface. There are many potential applications where the user wants a small amount of information, but a simple request/response protocol results in the transfer of large amounts of data. Large data transfers are undesirable both because of congestion on the Net and because of the trend for the Net providers to charge fees on a per-(mega)bit basis. These applications are better served by a stateful protocol which permits most of the processing to be performed on the server, only transmitting the relatively small result set of relevant items.

Second, those organisations mounting complex database applications are becoming concerned with revenue - either cost recovery for an intranet service or having the service on a commercial basis. One way to charge is per unit of resource consumed (the 0055 model), although some services charge a subscription fee for unlimited use (the cable TV model, Rushkoff 1996). Complex database services can permit queries

with a wide range of resource consumption, so are natural candidates for the 0055 charging model. The protocols used must therefore permit not only an accounting of resources consumed but also predictions of resource consumption framed as warnings to the client and requests for authorisation to proceed.

This paper first describes a number of examples of complex database applications, then extracts from them a set of communication protocol requirements needed to support them. It then argues that the Z39.50 protocol developed by the library community has already most of the features needed to support commercial access to text-type databases, and describes a number of proposed enhancements to the protocol to support the more powerful SQL query language, which permits access to a wider class of database and also improved services for text-style databases. The paper concludes with some ideas for further enhancement to the protocol.

Sample Applications

Typical Web sites provide access to databases in one of two ways. News sites (e.g. *The Australian*², *CNN*³) present a view of the current news, with hypertext links to more detailed and related, often older, items. Other sites permit the user to make a query, and construct a page on the fly from the result (e.g. the classified ad newspaper *Weekly Trading*

¹The work reported in this paper has been funded in part by the Cooperative Research Centres Program through the Department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

²<http://www.australian.aust.com>

³<http://www.cnn.com>

Post⁴, the on-line bookstore Amazon.com⁵, the resource discovery search engines). The former have small databases, while the latter have large databases but permit only very simple queries.

There is a long history of public access to large text-oriented databases, and many successful commercial enterprises. Library catalogs are often on-line. Many bibliographic databases distributed on CD-ROM are now being networked. Some newspapers offer on-line search of their archives (*The Sydney Morning Herald*, *The New York Times*). Legal databases offer both statute and case records (*CLIRS*). Finally, several industries, notably pharmaceuticals, could hardly exist without commercial chemical, pharmaceutical and medical information services.

These established services are being joined by a wide variety of others. Governments are putting huge amounts of information on-line (The City of Brisbane, The Queensland Government, the GILS project in the USA). Some of this information is presented as text-style databases, but there are also geographical databases, statistical databases and SQL databases, not to mention multi-media databases.

Most commercial information services pre-date the Web (for example the Dialog bibliographic service has been in operation for decades), and generally support a query system which is somewhat more complex than most present Web services. First, they support a query language which has different categories of text (author, title, abstract, keywords, etc.), and which generally permits boolean combinations of search terms (and, or, not). Second, most of the processing is done on the server, and the server maintains intermediate results which can be accessed by the client in later interactions in a session. Finally, many services allow the user to post queries against the stream of documents being added to the database (alerters), which accumulate results and periodically send them to the user, either on next log in or by some form of mail. Alerters are queries which reside in the server even though they belong to the user, so that the server maintains state across sessions.

Many of the new services noted above rely on the even more powerful SQL query language. SQL has the capability of the boolean query languages, but in addition can construct new tables by combinations of base tables, and support aggregation (count, sum, max, min). SQL is not the only language needed to support the range of new services. Geographical databases rely on a number of spatial query languages; statistical databases often have somewhat idiosyncratic query languages; and some multimedia databases permit search by non-textual content using a variety of languages. SQL platforms are evolving towards a more comprehensive object-relational language, which subsumes the specialised languages, including text

retrieval languages. This is one of the main thrusts of the SQL3 standards development (SQL3 1996) and is reflected in a number of major platforms including DB2, Informix and Oracle.

Web access would give wide availability to these new services, if only Web protocols could support the necessary query and accounting facilities. Further, if these more advanced facilities were available on the Web, there would be potential for additional new services and for existing services to be used in new ways.

Moreover, there is much scope for queries spanning many sites. For these to be successful there must be agreements on nomenclature over the sites. These agreements already exist in a few communities, notably libraries for bibliographic data, and in electronic commerce (Electronic Data Interchange or EDI, Becker 1990).

A particular example of a new class of service is 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 new types of queries on text databases, 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.

Requirements for a Protocol

In the previous section, we have argued that there are a large number of services loosely described as databases which would be advantageously placed on the Web, if only the Web supported adequate communication protocols. This section summarises the protocol requirements of these services.

Standardised complex query languages.

To gain maximum value from a service, the client must be able to support the query language employed by the server. There must be a small number of standardised languages, or it would be impractical for the client software to support them all; and even more important, it would be impractical for the user to understand them all. There are standards for the boolean query language and of course SQL. The SQL standard (SQL3) is being adapted to encompass other query languages such as spatial. This requirement includes standards for transmitting structured data result sets from the server to the client.

Ability to find metadata about services

A general problem in development of Web services involving complex data is how to describe the data.

⁴<http://www.tradingpost.com.au>

⁵<http://www.amazon.com>

This is the metadata problem. In particular, most SQL databases are accessed not with SQL directly, but via various kinds of user-friendly query builder software, either graphic or natural language based. In order for these query builders to work, they must know the structure of the target database. The target database must therefore be able to expose its structure. The SQL standard, for example, mandates certain structural tables which contain the schemas describing a database's contents.

Ability to exchange information about actual and potential resource consumption

Particularly with the more powerful query languages, there is a great variation in the amount of computing resource needed to support a query. In a commercial environment, it is essential for the server to be able to estimate the cost of a query, to transmit the estimate to the user, and to proceed only with the user's authorisation. Further, the final cost and the basis for its computation must be presented to the user via their client software.

Ability to maintain state throughout a session

In order for a service to change state with successive interchanges during a session it must be able to build data structures on its server associated with the session. The user must be able to enquire about the status of these structures and to be able to reference them in further queries. Finally, the server must break down these session-specific structures on termination.

Ability to maintain state across sessions

To support alerters and other cross-session state, a service must be able to identify a customer across sessions. It must be able to build data structures supporting its cross-session services, and support a user's queries on their status. There must be facilities to account for them and to present this accounting to the user.

Repositories of application-specific standard nomenclatures

In order for a query to be processed by several sites, it must be interpreted in the same way by all of them. Not only must the sites all support the same query language, but must agree on the interpretation given to the terms in a query. The organisations providing the sites must come to an agreement on nomenclature, and this agreed nomenclature must be published.

Such standard nomenclatures have been developed in the library and government information provider community, most notably for bibliographic reference but including also geographic and statistical nomenclatures. Many industries depend heavily on EDI which includes standards for many business documents such as requests for tender and invoices. EDI is emerging from its reliance on bilateral

agreements via the Open EDI standards (ISO 1994, Knoppers 1992).

Further, there is a proliferation of within-organisation and within-industry common SQL multi-databases (in the healthcare and travel industries particularly) which rely on universe-specific global schemas (Sheth and Larsen 1990).

To support easy Web-based access, these standard nomenclatures must be available in standardised repositories mounted as Web services themselves.

Broadcast/ multicast facilities

The applications requiring broadcast or multicast facilities are queries over multiple databases belonging to an association supporting a common nomenclature. Facilities required include identification of sites in the association, access to non-functional information including cost and availability, keeping track of which sites have responded to the query, cancellation of queries on sites which do not respond by a nominated time, and some way of combining results.

Z39.50 Facilities

The library and government information communities have been early implementers of database applications such as we are concerned with, in particular for library catalogues and other bibliographic databases. A communications protocol standard called Z39.50 has been developed by this group of people under the auspices of the American National Standards Institute. The protocol is now in its third version (ANSI 1995) and has been implemented in many organisations, particularly libraries, including The University of Queensland Library in a Web environment. An earlier version of Z39.50 was the basis for the WAIS (Wide Area Information Server) database network (Marshall 1992). This section is an overview of version 3 of that protocol. More detail may be found in the appendix.

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 the client and the server communicate and interoperate even when there are differences between computer systems, search engines, and databases.

A series of messages passing between the client and server (defined by what the standard calls the Initialization Facility) establish a connection, initiate a Z39.50 session (called a Z-association), and negotiate expectations and limitations on the activities that will occur (e.g., maximum size of the records that will be transferred from the server to the client, the version of the protocol supported, etc.). After these agreements are negotiated, the client may submit a query. The Z39.50 client translates the query into a standardized representation and passes it to a Z39.50 server (defined by the Search Facility). The server executes the search against a database(s), and a result

set is created. The client can then ask for records from the result set or request from the server additional processing of the result set (defined by the Retrieval Facility). Upon receipt of the records, the client may process the records and display records to the user. The extent to which a client can perform additional processing on retrieved records (e.g., combining records from several separate searches) will depend on the user interface software (Moen 1997).

The standard at present specifies six query types, representing a variety of ways to search textual databases. Of these, Z39.50 fully specifies and mandates support of the boolean query language typical of library catalogues and CD-ROM bibliographic databases. The others are listed in the appendix.

The standard includes a facility called standardized attribute sets. If this facility is employed, the attributes associated with a search term belong to a published attribute set, defining the attribute semantics for a given domain, ie 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 listed in the appendix.

Response Records from the server include database and diagnostic records. Both record types may be returned in several formats, which like attribute sets are also registered. Formats supported are described in the appendix.

Z39.50 Version 3 provides eleven facilities, described in detail in the appendix, which satisfy the first five of the seven requirements listed in the previous section for a protocol to support complex database services on the Web. Z39.50 satisfies these for text databases, an important class of problem. As to the last two requirements, Z39.50 has application-specific standard nomenclatures, but they are generally implemented hardwired into the user interface. (There is only one - bib-1 - in wide use.) Similarly, there are clients which support a broadcast query, but generally over a specific group of services and using techniques idiosyncratic to that group.

Zinc Enhancements

Z39.50 meets many of our requirements for a complex Web application protocol. However, it is designed around text database applications and therefore has a limited query language. Although the Z39.50-supported query languages can be used on SQL databases, full SQL queries allow much more functionality. The functionality is being further increased in SQL3. Our Zinc project extends the Z39.50 protocol so that full SQL query and retrieval can be supported.

Structured organizational databases, typified by SQL databases, at present are rarely available on the public networks. Instead, their networks are closed, either limited to an organizational environment or proprietary products. They network in two ways, client-server and multidatabase, via middleware. In both cases, knowledge of system catalogues and query language dialects must be hardwired into the application or the middleware platform.

The technical problems of making an SQL database available on an open network environment are analogous to those of making a text database available on that environment. Firstly, the server must have a standard way of making its system catalogues available to the client, of agreeing on the query language dialects and types of data available, and a standard means to verify version compatibility between client and server. Secondly, the query language must support the ability of the server to not only process a query but to retain it as the basis for further queries, in the same sort of way as managing saved result sets in text databases. Finally, the server must support a standard means to manage and account for software actions (such as triggers) originating at a client but residing at a server site. All of these facilities are available in the Z39.50 protocol; the Z39.50/SQL+ project presents an adaptation of that protocol to an open SQL environment.

Z39.50/SQL+ can be seen as an extension of the existing Z39.50-1995 (Version 3) protocol, uniting the advantages of an SQL RDBMS with those of Z39.50. This SQL extension is most beneficial (but not restricted to) when a client wishes to manage information retrieval from SQL databases.

Z39.50/SQL+ provides to the SQL user a stateful communication environment with the full flexibility and query power of SQL when connecting their working environment to a remote database. Z39.50/SQL+ clients will be able to formulate complex queries, either by using SQL or one of its derivatives, such as Query-by-Example (QBE). Queries may be formulated on multiple tables supporting cartesian products, unions, intersections, joins on matching columns, and projections on given columns, as well as being able to use powerful constructs for expressing conditions, performing aggregate and comparison operations, partitioning tables into groups, among others. In addition, SQL RDBMSs provide structured, organizational databases complete with data management facilities including system catalogues, flexible indexing and query optimization - providing efficient access and retrieval of both data and metadata.

Z39.50/SQL+ introduces a new query type, a type-SQL query - conforming to the SQL3 Standard. The SQL query expression is highly structured, allowing search terms and attributes to be specified within the query. The results of the SQL query would be stored in a named table.

The Z39.50 standard attribute sets are essentially the same things as relational database schemas, providing a natural way to integrate the standard nomenclatures and ontologies being developed into an SQL-capable information service.

However, it should be noted that a service need not rely on a standard attribute set, as the SQL server database already stores its schema metadata within its system catalogues. These catalogues could be accessed via the Explain facility for dynamic building of client interfaces.

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 RDA (Remote Database Access, RDA 1993) with support for most SQL3 datatypes.

No changes to the resource report and access control formats are envisaged at this stage. Minor extensions to the Explain record syntax include SQL version, database version and catalogue table name parameters. (See appendix for descriptions of these facilities.)

Z39.50/SQL+, like RDA and ODBC (Geiger 1995), uses SQL as its query language. However, it should be noted that the functionality of Z39.50/SQL+ is conceptually different from either of these. Both RDA and ODBC 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 managing interoperable information retrieval. It facilitates the management of state within and across communication sessions, provides a standard means of dynamically accessing the RDBMS catalogues, and the option of using a standardised data model for interoperability, as well as extra management facilities such as access control, resource control, scan, extended services etc.

Z39.50/SQL+ is intended to expand the usage of Z39.50 away from the library community, which is mostly text based, to a much broader community employing relational databases.

Furthermore, relational databases, at present, have very limited and rudimentary web access. With the escalating usage of the web - for sophisticated access to catalogues, data warehouses and the like - Z39.50/SQL+ offers a higher, management-level information retrieval protocol.

The main advantages over existing RDBMS middleware are

- Platform-independent interoperability
- Maintenance of state on the server within and across sessions - allowing queries, result set(tables), and

triggers to be stored on the server either temporarily or permanently

- Predefined global schemas (attribute sets) for specialized communities allowing broadcast queries/merging of results from multiple databases as well as additional facilities for the management of information retrieval
- Scan - browse-like facility
- Continuous segmented transmission of large result sets
- Dynamic orientation to previously unknown databases (Explain Services)
- Document/item ordering (Extended Services)
- Update facility (Extended Services)
- Server-initiated Access and Resource Control (including Accounting Services)
- Operation cancel (e.g., Search)

Our modifications to Z39.50 extend the facility satisfying the first five requirements to another large and important class of database.

Implementation

Zinc is the prototype being built to demonstrate the functionality of the Z39.50/SQL+. Zinc offers a management-level services for information retrieval incorporating SQL searches on remote databases over the Internet, and has the potential to search multiple heterogeneous databases in an interoperable way, merging structured result sets for display or manipulation. It maintains and manages state on the server database. Zinc will be incorporated into the HotOIL prototype⁶ for finding resources which are distributed across heterogeneous networks.

A sketch of the implementation is given in Figure 1. Netscape is used as the Internet Browser. An HTML form gives access to the Z39.50 Service, which presents a dynamically created QBE-style interface from which the user may define a search or present request. On sending the search request, the client-side user interface then hands control over to the Z39.50 Client/API which sends the constructed data packet to the Z39.50 Server/API. The Z39.50 client and server are based on the Z39.50 YAZ toolkit⁷. The Z39.50 Server/API handles the request by means of a backend server, which in turn, translates the request into a command recognizable by ODBC, making a dynamic call to the server database (in our case, Oracle). On the response to a request the order of events is reversed. The results of a search/present request, are then

⁶

<http://www.dstc.edu.au/RDU/APAP/HotOIL/HotOIL.html>

⁷ <http://www.indexdata.dk/yaz.html>

RDA (1993) ISO/IEC 9579.2 1993 PDAM1 Information Technology - Open Systems Interconnection - Remote Data Access Protocol Part 2 - SQL Specialization - Amendment 1 Support for SQL 92.

Rushkoff, D. (1996) "Gated access: a bad sign" The Australian December, December, p. SYTE 3.

Sheth, A.P. & Larsen, J. (1990). Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases, ACM Computing Surveys: Special Issue on Heterogeneous Databases 22(3) 183-236.

SQL3 (1996) ISO/IEC 9075-2: JTC1/SC21 N10489:1996 Information Technology- Database Language SQL Part 2: SQL Foundation.

Appendix: Z39.50 Details

Z39.50 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 Interactive 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

Of these, Z39.50 fully specifies and mandates support of the type-1 query (2).

Standard attribute sets supported at present are:

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

(Explain and Extended Service are meta facilities associated with the protocol, described in more detail below.)

Response Records formats supported include:

SUTRS - Simple Unstructured Text Record Syntax (text only)

GRS1 - Generic Record Syntax, to return records with structure

Explain - Server Information syntax

ESTask Package - Extended Services record syntax

OPAC - Online Public Access Catalogue

Summary - Bibliographic Summary syntax

MARC formats - USMARC, UNIMARC, UKMARC, and CANMARC

and a number of diagnostic error formats.

Z39.50 Version 3 provides eleven facilities:

Initialization Facility -- allows the client to negotiate a Z-association.

Search Facility -- enables an client to query databases at a server system, creating a result set of records on the server and to receive information about this result set.

Retrieval Facility -- (two services)

Present: allows the client to request one or more records from a specified result set.

Segmentation: if the records requested by a present request will not fit in a single segment, and if segmentation is in effect, the server returns multiple segments, each of which contains a portion of the records.

Result-set-delete Facility -- enables an client to request the deletion of specified result sets, or all result sets.

Access Control Facility -- allows a server to challenge an client. 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 -- (three services)

Resource-Control: permits the server to send a resource-control request - eg notifying the client that either actual or predicted resource consumption will exceed agreed upon limits, and request client consent to continue an operation

Trigger-resource-control: permits the client to request that the server initiate the resource-control service, or cancel the current operation.

Resource-report: permits the client to request a resource-report pertaining to a completed operation or to the Z-association.

Sort Facility -- allows an client to request that the server sort a result set (or merge multiple result sets and then sort).

Browse Facility -- used to scan an ordered term list (subject terms, names, titles, etc.).

Explain Facility -- Used to discover details of the server 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.

Extended Services Facility -- provides access to services outside the protocol, which may persist after the Z-association has been terminated. The extended services defined by this standard include persistent result sets, persistent queries, periodic search schedules, item order, and database update.

Termination Facility -- allows either an client or server to abruptly terminate all active operations and to initiate termination of the Z-association.