The Implementation of Z39.50 in the National Library of Canada's AMICUS System

J. C Zeeman
Software Kinetics Limited.
Stittsville, Ontario
zeeman@sofkin.ca

Abstract

AMICUS is the National Library's new integrated bibliographic system. The initial phase of development, released in the second quarter of 1995, supports cataloguing and catalogue products, bibliographic searching and customer information management. The search module is implemented as a Z39.50 server that accesses the two database engines integrated into AMICUS: a relational database for bibliographic data management and a full text database for keyword searching and future full text access. An overview of the AMICUS applications is followed by a brief introduction to the modelling of AMICUS bibliographic information in the relational database. A more detailed description of the architecture of the AMICUS search engine is given, describing components, internal messaging, query analysis, optimization, semantic mapping and record conversion. A description of the three AMICUS clients concludes the paper.

Background

The National Library of Canada is the national copyright deposit library in Canada. It serves as the primary cataloguing agency for Canadian published materials and as the national agency for assignment of International Standard Bibliographic Numbers (ISBNs) and International Standard Serial Numbers (ISSNs). It makes catalogue records available to a large number of Canadian libraries and exchanges national level records with both the Library of Congress and the British Library to make their records available as source records in Canada. The NLC maintains a large union catalogue of Canadian holdings and hosts the catalogues of several other federal agencies in Ottawa (the "full-service libraries"), notably the catalogue of the Canada Institute for Scientific and Technical Information (CISTI) — the library of the National Research Council of Canada. Because Canada is a bilingual country, cataloguing is done in both English and French, and the NLC offers its products and services in both languages.

The NLC began preparations for the replacement of its bibliographic system, DOBIS, in the late 1980s. A fundamental conclusion arising from extensive investigation was that no vendor could provide an off-the-shelf product that would meet the complex requirements of the National Library in terms of support for multiple languages, standard number assignment, and multiple overlapping databases to support:

- Canadiana cataloguing,
- multiple sets of source records for both bibliographic items and authorities,
- the Canadian union catalogue incorporating records of highly variable quality, and
- the individual library catalogues of the National Library and its full-service partners.

In 1992 a Canadian systems integrator, Groupe CGI, was competitively awarded the contract to develop a new information system for the National Library. Software Kinetics Limited teamed with CGI to develop the winning proposal and acted as subcontractor during design and implementation of the system, with particular involvement in the design and development of the bibliographic search engine.

.

Amongst other requirements the RFC specified that the system must:

- be based on a relational database system(RDBMS) for management of bibliographic data;
- support keyword searching;
- meet stringent performance requirements for both cataloguing and searching;
- support Z39.50 access; and

- support an initial database in excess of 10 million records, rising to 20 million over the life of the system.

The winning bid proposed the use of Digital Equipment's VAX hardware platform with the VMS operating system, the Ingres RDBMS and Fulcrum's Ful/Text engine for keyword access. Catalogue access would be provided for a Windows graphical user interface (GUI) client and for a host-based terminal client. Analysis, design and as much of implementation as possible would be done using CASE tools.

Work on the project plan began in July 1992. Development was largely completed by April 1995 and the system went into production on 12 June 1995.

AMICUS applications

Cataloguing

Cataloguing is implemented as a Microsoft Windows-based client-server application developed using the Ingres OpenRoad application development tool. Cataloguing is intimately connected with searching: the cataloguing functions are available to authorized users as menu options from the search windows and vice-versa. The main cataloguing window appears as a MARC-based worksheet. Cataloguers can copy-catalogue from the large set of source records; edit previously created records either to correct errors or to create new records for similar items; or add new records.

The basic worksheet is a blank form on which the cataloguer can create a MARC-tagged record. The same form is used for all record types, including authorities. The cataloguer can input tags, indicators and codes directly, from memory, or can choose appropriate values from labelled pop-ups. All input is validated and updates the database immediately.

The cataloguing application matches cataloguer input for controlled headings, including names, titles, subject headings, control numbers, classifications and call numbers with data elements existing in the database. If the cataloguer's input is matched, the access point table for the appropriate heading is automatically updated. If the input is not matched, the cataloguer may browse existing values in the database to find the appropriate term. If a term is not found, the cataloguer may choose to add the term to the database, in which case he or she is presented with a worksheet on which to enter the required control information. When this information is supplied, the original workform is automatically updated with the new data element.

Standard Number management

The National Library acts as the Canadian numbering authority for International Standard Book Numbers (ISBNs) and for International Standard Serial Numbers (ISSNs). To facilitate its management obligations, AMICUS incorporates applications that allow National Library staff to manage the assignment of number blocks to publishers and trace number assignment to individual items.

Bibliographic Searching

Bibliographic searching is supported for users of the AMICUS Windows-based client and for users of terminals connected to the host VAX computer via telnet or Datapac, the Canadian public X.25 network. For these latter users, two host-based search interfaces have been implemented: a command-based search interface for experienced searchers, modelled on the NISO Z39.58 Common Command Language; and a form-based interface for patrons of the National Library and CISTI reading rooms and reference services. All these interfaces use the NISO Z39.50 Information Retrieval protocol to communicate with the bibliographic search engine located on the VAX computer.

The bibliographic database is maintained as a fully normalized Ingres relational database, with keyword indexes to bibliographic records maintained in the Ful/Text component of the system. The bibliographic search engine manages concurrent searching of both database systems transparently from the user's point of view.

The AMICUS system allows the user to search a number of databases. At present the following databases are available:

- Canadiana bibliographic records,

- Canadiana authority records,
- Library of Congress Cataloguing Distribution Service bibliographic records,
- the catalogue of the National Library's collections,
- the catalogue of the CISTI collections,
- the catalogue of the AMICUS Full-Service libraries,
- the Canadian Union Catalogue.

In addition the user can search the combination of all the bibliographic databases as "Any AMICUS database".

These databases are implemented as logical subsets of the single AMICUS physical database. The same physical database record can be associated with multiple logical databases and different, possibly conflicting, data elements can be part of the record in different databases.

User/Supplier Information

AMICUS includes applications to manage information pertaining to the users of National Library systems and information and also to the suppliers of information to the National Library, including publishers participating the cataloguing-in-publication, ISBN and ISSN programs and libraries contributing to the Union Catalogue.

Products

The existing set of National Library bibliographic products, including *Canadiana*, the national bibliography, will be produced from AMICUS. The system includes applications to generate and manage these products.

Billing

The National Library has an obligation to recover a portion of its costs and most of the bibliographic services provided by the National Library to other libraries are therefore charged services. AMICUS includes applications to monitor usage and generate billing information for the production of invoices.

Functions Not Implemented

The current release of AMICUS does not support the following functions, among others:

- circulation
- acquisitions and serials control
- ILL messaging
- financial management
- publishing and ad-hoc products

These functions are currently provided by legacy systems or by the Dynix integrated library system acquired by the National Library to serve as an interim measure until the functions can be integrated into AMICUS.

The Relational Database

AMICUS is fundamentally a relational database. This has a significant impact on how searches are performed in the system.

A greatly simplified version of a portion of the logical data model used for AMICUS bibliographic information is shown in Figure 1 below. Arrows in the diagram point in the direction of the "many" aspect of a one-to-many relationship. The diagram shows how some of the entities in the system are related to each other. To keep the diagram simple many entities, such as subject headings, notes, physical descriptions, location information, etc. have been omitted. It should be noted how names, titles, etc. are related to bibliographic items and to authorities and also how bibliographic items are related to physical copies and to each other. There is a separate access point entity for each entity that has a many-to-many relation with the bibliographic item. Most of the relationships shown in the diagram are optional in the sense that not every instance of the entity will have a corresponding access point: for instance, some bibliographic items will have no name heading related to them.

Each of the entities has a number of attributes that specify both the data elements that form the entity and the entity's relationships with other entities. The principal attributes for the three major types of AMICUS entities are shown below.

Principal attributes of a headings entity:

- heading number
- heading type (e.g. "personal name inverted order" for a name, etc.)
- heading display text
- heading searchable sort form
- heading inverse sort form
- control information (e.g. language, verification level, etc.)

The "heading searchable sort form" is a normalized form of the heading to allow the headings in the table to be sorted in lexical order, for production of scan lists, and to enable a heading be matched against a searcher's input term without having to worry about variations in capitalization and punctuation. This normalized form is stored as a distinct data element, and is in fact the principal element used for searching. The exact nature of the normalization performed depends on the heading type. Typically punctuation and MARC subfield codes are removed, all text is converted to upper case, non-ASCII characters are mapped to ASCII equivalents, etc.

The "heading inverse sort form" is present to allow backwards movement in a scan list. ISO standard SQL does not support moving backwards in a table, so the inverse form contains a simple binary inversion of the searchable form, which has the effect that moving forwards in the order of the inverse sort form actually moves backwards in the order of the searchable sort form.

Principal attributes of an access point entity:

- heading number (to provide access to and from rows in the headings tables)
- bibliographic item number (to provide access to and from rows in the items table)
- heading function (e.g. "added entry", etc.)
- control information

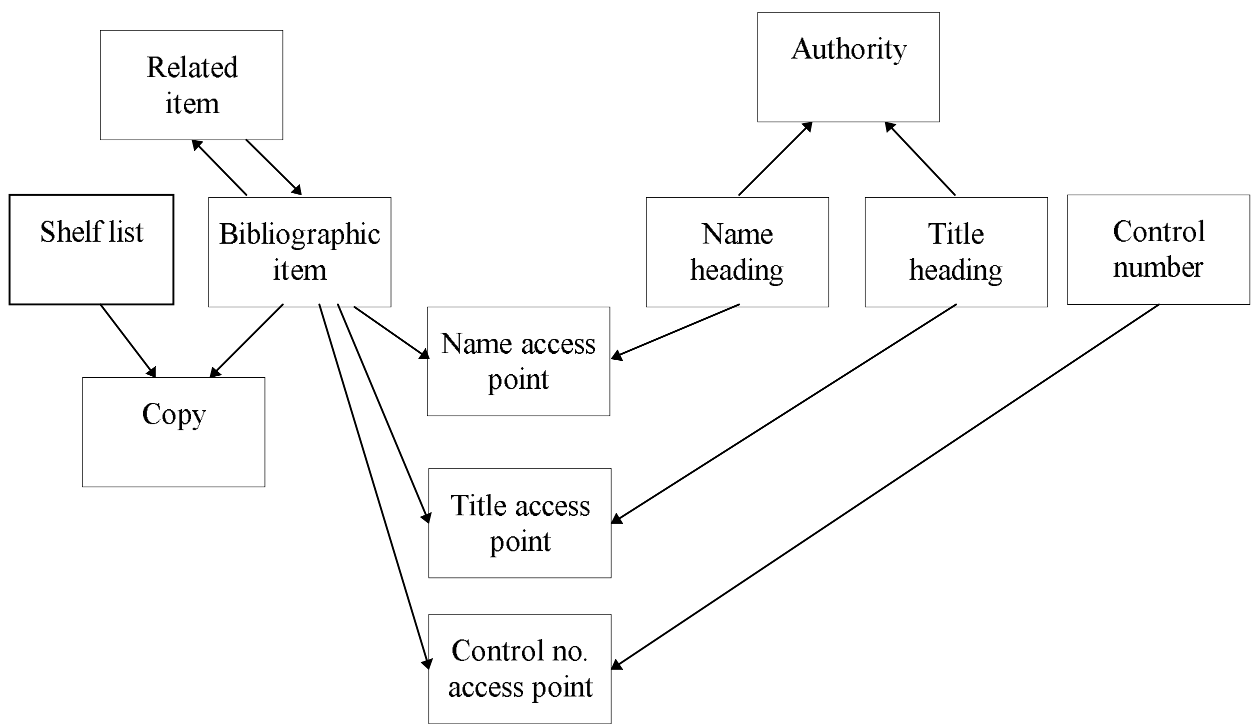

Figure 1 - Portion of AMICUS Logical Data Model

The heading number and bibliographic item number attributes allow instances of the bibliographic item entity that are associated with a given heading to be selected and vice versa. The heading function element allows appropriate displays to be constructed on the basis of the heading function and also allows selections to be made on the basis of the type of relationship between the heading and the bibliographic item.

Principal attributes of the bibliographic item entity:

- bibliographic item number
- various coded elements (e.g. bibliographic level, record type, country of publication, etc.)
- date of entry on file and of last transaction
- dates of publication
- control information.

It should be noted that the bibliographic item entity contains none of the descriptive information or headings normally considered to be part of the "bibliographic" information.

This data model is used as the basis for the physical design of the database. The database consists of a series of tables, each corresponding to a single entity in the logical model. The attributes of the entity become the columns in the table and the data records become rows. Any individual element in the database can therefore be identified as the intersection of a row and column in a particular table. The internationally standardized Structure Query Language (SQL) is used to create the database and to access and manipulate all the information in the database, and forms the only valid means of accessing the data in the database.

An SQL statement to retrieve bibliographic records related to a name would be as follows:

```
SELECT
        itemNumber
FROM
        nameHeading,
        nameAccessPoint
WHERE
        nameHeadingSearchForm IS LIKE
            'JOHN SMITH %'
        AND
                nameHeading.nameHeadingNumber =
                    nameAccessPoint.nameHeading-
                    Number
```

This query selects values of the item-number column in rows in the name-access-points table that are identified by having the same value in the name-heading-number column as rows in the name-headings table that are identified by having a value in the name-heading-sort-form column beginning with the string "JOHN SMITH ". It should be noted that it is not necessary to access the bibliographic items table in order to create a set of bibliographic item numbers.

An SQL statement like that shown above is readable by a human operator. It represents, however, a very simple case. When additional qualifiers are added, such as heading types and heading functions, SQL statements quickly become too long to be easily input or even understood by a human operator. System users cannot be expected to routinely use SQL to query the database and SQL is in fact normally generated by an application that presents a more intuitive interface to the user and shelters him or her from the physical schema of the database and from the complexity of the SQL syntax.

Although a large number of tables can theoretically be joined in one query, experience has shown that performance begins to degrade rapidly when an SQL Select statement requires accessing more than three tables. The AMICUS data model allows single-term searches to be executed with only two-table accesses in all but a very few instances; for these, three-table accesses are required. Four-table or more accesses are never used in the AMICUS bibliographic search engine.

The Full Text Engine

Keyword access to AMICUS bibliographic data is provided by use of a separate database system — independent from the relational DBMS — the Ful/Text DBMS. This system maintains keyword indexes of all names, titles, subject headings and bibliographic notes. Searching is performed via a proprietary Application Programming Interface (API) that supports Boolean operations, proximity searching, thesaurus look-up and substitution, and relevance ranking of results. The last two features are not used in the current AMICUS implementation, although they will offer considerable power in later phases of development when full text and other data is added to AMICUS.

A Ful/Text database is modelled as a set of documents that are searchable via the engine. To allow specificity of searching, a document can be divided into a number of “zones” that can be searched individually or in combination. For AMICUS records the following zones have been defined, “author”, “title”, “subject”, “publisher” and “notes”. Zones in a Ful/Text query are specified using numeric zone identifiers.

Ful/Text does not require the documents themselves to be stored as part of its database. Documents can be external entities to which Ful/Text maintains searchable indexes together with a catalogue of pointers back to the documents themselves. AMICUS thus maintains no bibliographic data in the Ful/Text database. Instead, the Ful/Text indexes refer to the bibliographic records in the Ingres database.

The simplest Ful/Text query consists of a single word to be found: e.g., “SMITH”. This will build a result set of documents containing the word “SMITH” anywhere in the document. Searches can be restricted to specific parts of documents by using a “zone operator”: to find a word used as part of an author’s name, the search string would be “\C40s SMITH”, where “\C” represents an escape sequence, “s” is the zone operator and “40” the parameter indicating the zone value to which the search is to be restricted. More complex queries are created by adding Boolean, proximity and other operators.

In addition to the zone operator, Ful/text supports a number of other operators that act on multiple words. The syntax of these operators requires a leading escape (represented by “\C”), one or more optional parameters, the operator identifier, the words the operator is to be applied to and a final escape following the term plus a closing brace (“\C}”) to indicate the limit of the scope of the operator. The proximity operator, for example is “\C<distance>p ... \C}”. A phrase is searched as two adjacent words by using the proximity operator with a distance parameter of zero, so the Ful/Text query for a phrase used in an author’s name is: “\C40s \C0p SMITH JOHN \C}”. Operators can be nested and can occur in any order.

The basic search unit of Ful/Text is the word, and all operators other than proximity apply to one or more words. The only distance unit supported for proximity is character distance. AMICUS therefore, also supports only the character unit for proximity distance.

The result of a Ful/Text search is a result set of document ids maintained by the Ful/Text engine. The bibliographic search engine obtains these ids from a Ful/Text search to use as either an interim result set or as the final result set of the search as appropriate.

The Bibliographic Search Engine

The bibliographic search engine is at the heart of the AMICUS bibliographic information management and retrieval functions.

The principal requirements to be met by the AMICUS search engine were as follows:

- to integrate search access of the relational and the full text databases;
- to integrate with the cataloguing application being developed using Ingres Windows-4GL;
- to support a very large database;
- to provide a Z39.50 version 2 target;
- to support up to 250 simultaneous users;
- to meet stringent performance requirements.

The design approach was to modularize searching as much as possible into separate components, each dedicated to a specific role in executing the Z39.50 query and each operating independently and simultaneously. The general architecture of the search engine is shown in Figure 2 below (acronyms are explained below).

The search engine thus consists of a number of processes that together implement the search functionality supported by AMICUS. Each of the processes in the search engine operates independently of all others, and uses asynchronous messaging to communicate with the other processes as needed. Each process maintains its own message queue and deals with each message in the queue in turn. While the message queue is currently implemented using the VMS mailbox utility that provides efficient low-level support for interprocess communication, message handling is sufficiently isolated that a different messaging mechanism, such as RPC, could be implemented without major disruption should the engine be transferred to a different platform.

Each of the processes is dedicated to a specific task in responding to a Z39.50 protocol message.

EIT The *External IR Target* implements the Z39.50 protocol machine for non-AMICUS users. It is based on the IR Toolkit software developed by Software Kinetics Ltd. for the National Library of Canada. It implements version 2 of the protocol as specified in the 1992 standard. All services of version 2 are currently supported with the exception of AccessControl, DeleteResultSet, and ResourceReport.

The EIT polls for incoming Z39.50 Application Protocol Data Units (APDUs). When an InitRequest APDU is received, it decodes the APDU and issues a message to the MISR process (see below) requesting validation of the user's authentication information and the supply of session information such as the various resource limits to apply to searches and the set of databases the user is allowed to search. The protocol machine uses the response from the MISR to formulate the InitResponse APDU.

APM The *AMICUS Protocol Manager* process implements a proprietary protocol developed for use between AMICUS clients and the bibliographic search engine. This protocol acts as a wrapper around Z39.50 protocol messages and other messages used by AMICUS clients. AMICUS clients offer additional search services not available to external Z39.50 clients, such as saving queries and result sets. These services were not available in the 1992 text of Z39.50 and what became the 1995 specifications were not sufficiently stable when the AMICUS design was finalized in early 1993 to allow them to be implemented. Therefore, a proprietary protocol has been used.

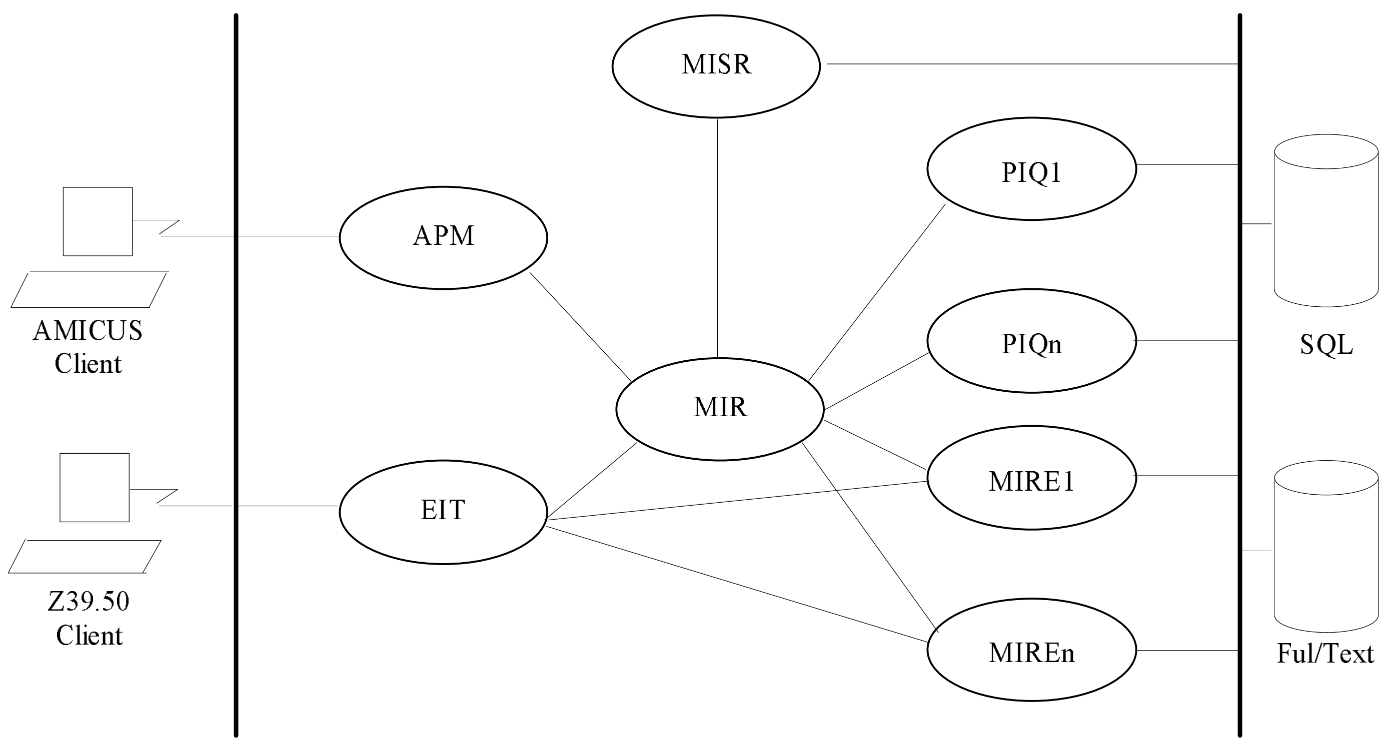

Figure 2: Bibliographic Search Engine Architecture

The AMICUS protocol is also used by AMICUS clients to request that a MARC record image be placed in an Ingres table for subsequent use by the client. The AMICUS protocol is furthermore designed to permit the transfer of Z39.50 messages to and from a remote Z39.50 host via a client gateway on the AMICUS server. None of these services are available to non-AMICUS Z39.50 clients.

The APM includes the same Z39.50 protocol machine as the EIT and generally behaves in the same way. Principal differences lie in the internal naming of result sets and in the creation of records. AMICUS clients do not use Z39.50 to obtain records for display, but instead manage their own displays directly from the Ingres database.

MISR The *Manage IR Security and Resources* process finds the message in its queue and executes the appropriate SQL statements to obtain authorization and session information. If the user has supplied a valid user id and password the MISR obtains the necessary session information, such as resource limits that apply to the user and databases the user is allowed to search, and makes it available to other processes for use as required.

MIR Query analysis, optimization and execution is managed by the *Manage Information Retrieval* process. There is a single MIR process that continuously loops through all outstanding searches, analyzing queries and dealing with messages from other processes as necessary.

The *query analyzer* receives a decoded query from the protocol engine (APM or EIT) and builds a *query execution tree* based on the logic of the query. Figure 3 below illustrates the query tree that would be built from the common command language query “find TW cats or dogs and SU pets or animals and DA > 1993” (find records with title word “cats” or “dogs” and with subject heading “pets” or “animals” and with publication date greater than 1993). As illustrated in Figure 3 below, each term in the query becomes a leaf of the tree and leaves are joined into branches with the operators in the query. Execution of the query begins with the bottom left-hand leaf and proceeds upwards and to the right. Branches of the tree may be optimized to make most effective use of the database engines' native query optimizers while minimiz-

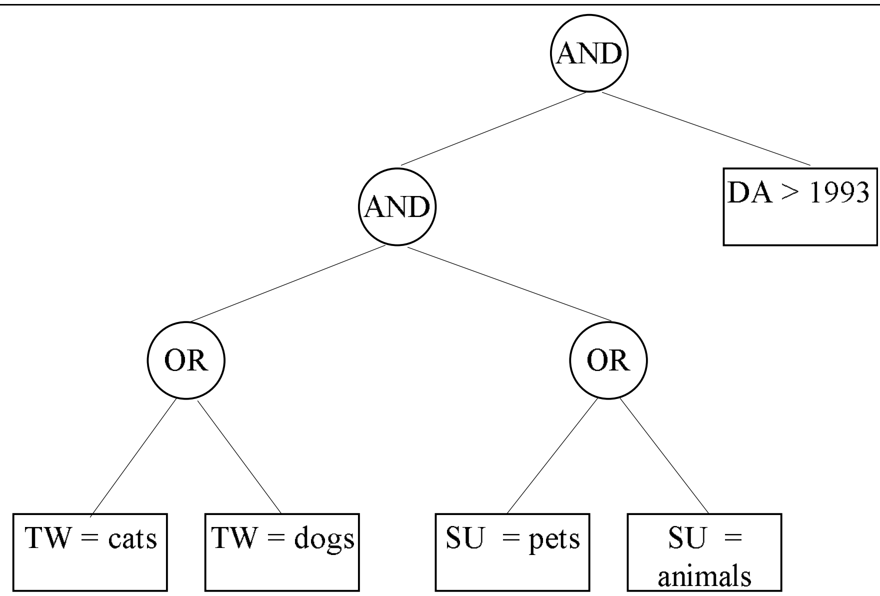

Figure 3 - Query Execution Tree

ing query execution time. In this query, the title-word terms are passed to the Ful/Text engine as a single subquery, and the subject-heading terms are passed to the Ingres engine as a single subquery using the SQL "union" operator. The result of each search is an Ingres table holding the intermediate result. These two subquery results are next joined into a single intermediate result and, since date of publication is not indexed in the Ingres database, the intermediate result will finally be joined with the date attribute in the bibliographic items table to form the final result set.

Query execution is implemented as a finite state machine that makes recursive passes through the query tree, optimizing where possible, dispatching subqueries to a database interface process (a PIQ, see below), interpreting results and changing the state of the various nodes as appropriate. Each query node may be in any one of the following states: "incomplete", "wait for dependent", "wait for PIQ", "wait for result", "complete". The query is reprocessed until the top-most ("root") node reaches the "complete" state at which point the query has been fully processed and the result set (if any) has been built.

When the MIR receives a message from a PIQ that a subquery has been processed, it finds the node associated with that PIQ and changes the state of the node as appropriate. It then reanalyzes the query, changing the state of other nodes as necessary. For instance, if the dependents of a query node have all reached the "complete" state, the node can then be changed from "wait for dependent" to "wait for PIQ", at which point the subquery specified by the node will be executed.

MIRE The *Manage IR External* process manages the creation of MARC records for return to an external Z39.50 user. There are multiple MIRE processes active simultaneously; the number is specified at search engine startup time, allowing system managers to tune this number for optimal use of system resources and performance. If the SearchRequest contains a "piggy-backed" present, the MIR will ask the next idle MIRE to create one record required for the response. If multiple records have been requested, multiple requests will be passed to one or more MIRE processes. If the EIT receives a PresentRequest it will repeatedly ask the next idle MIRE to create the next record required for the response. To create the record the MIRE executes a database procedure that extracts the required data from the various database tables and then assembles this data into the MARC exchange format. At present only the CanMARC format is supported for output.

PIQ All search interaction with the two database engines is handled by a *Process IR Query* process. As with the MIREs, there are multiple PIQ processes active simultaneously, with their use managed by the MIR. The number of simultaneous PIQs is specified when the search engine is started. Each PIQ can interact with both the RDBMS and the full text engine. Since execution of SQL statements by the RDBMS is synchronous (i.e. the process blocks until the query completes), using multiple PIQs allows multiple subqueries to be executed simultaneously.

The PIQ executes the database interaction and builds either an intermediate or a final result set. These sets are placed in RDBMS tables. Each PIQ processes a single subquery, which may be an SQL select statement to evaluate a single term, an optimized query that contains several terms, an SQL statement to perform a Boolean operation on the results of previous terms, or a full-text query.

Full-text queries are always optimized as much as possible. The result of a full-text query is a list of matching bibliographic item keys, which is copied into an Ingres table as the final or an intermediate result set.

Semantics Tables

The query analyzer has no built-in knowledge of the semantics of the AMICUS databases or of an AMICUS search. All the semantic knowledge relating to the physical data model of Ingres and the Ful/Text database structure resides in separately maintained *semantics tables* used by the query analyzer to generate subqueries for execution by a PIQ. There is one semantics table for each record type in the system that requires different semantic processing. In AMICUS at present there are separate semantics tables for authority records and for bibliographic items. Adding search support for other record types may involve as little work as defining a new semantics table.

The query analyzer has a only basic understanding of how to construct an SQL statement that involves a two- or three-table join and how to construct a Ful/Text query. It also incorporates a growing set of deterministic optimization cases based on comparison of tables names and other data from the semantics tables.

The semantics tables are held as Ingres tables for ease of maintenance, but for performance reasons are stored in memory while the search engine is running. This permits the search semantics to be altered in the database tables as required, without interfering with the operation of the search engine. Alterations will take effect the next time the engine is restarted (normally daily).

The semantics table for a database lists every Z39.50 Bib-1 attribute combination supported for that database, and, for each, specifies the semantics of the subquery that corresponds to the attribute combination. Some of these combinations specify full-text searches, others specify SQL searches of the Ingres database. For every term in a query the attribute combination is looked up in the table. If a row is not found, an "unsupported attribute combination" diagnostic is returned and the search is failed. If the row is found the query elements are placed in a memory structure used by the MIR to construct a subquery that is passed to a PIQ for execution.

Each row in a semantics table contains the following information:

- the attribute combination,
- parameters used in generating the searchable form of the term,
- a flag to indicate whether the attributes specify an indexed or unindexed database element, and either
- a skeleton Ful/Text query or
- the components of the SQL query.

The elements of the last item are used to create an SQL select statement for the subquery and may be used to optimize several subqueries into a single more complex select.

For example, the attribute combination:

> use = 4 (title),
> position = 1 (first-in-field),
> relation = 3 (equality),
> structure = 1 (phrase),
> truncation = 100 (do not truncate),
> completeness = 1 (incomplete subfield)

retrieves the following parameters from the semantics table:

> primary search table = titleHeading
> first join table = titleAccessPoint
> second join table = null
> key of primary table = titleHeadingNumber
> join table key for first join = titleHeading-Number
> join table key for second join = null
> element name from which to select record id = bibliographicItemNumber
> element name in primary table in which to match term = titleHeadingSearchForm
> SQL operator = LIKE
> variable string containing the search term = '%s %%'
> string constant to add to the SQL statement = null

These parameters lead to the creation of the following SQL statement (assuming the user's search term is "Rape of the lock":

```
SELECT bibliographicItemNumber
FROM titleHeading, titleAccessPoint
WHERE
        titleHeading.titleHeadingNumber =
titleAccessPoint.titleHeadingNumber
AND
        titleHeading.titleHeadingSearchForm
IS LIKE 'RAPE OF THE LOCK %'
```

Note that the C-language "sprintf()" function is used to place the term into the SQL statement. This function replaces a "%s" in the input string with a variable (in this case the search term). To use “%” as a literal character in the output string requires that it be repeated in the input string, thus “ ‘%s %%’ ”. The single quotes are literal characters in the output string that are required by the SQL syntax.

Most of the data in a bibliographic record (in particular, almost all coded data elements in a MARC record) can be searched using the search engine.. To support searching these coded elements, some 240 local use attributes have been added to the basic Bib-1 attribute set. Not all these elements are indexed, however. If these unindexed elements were searched on their own, complete Ingres table reads would be required to satisfy the query, a process which would take hours, if not days, in a database of 20 million bibliographic records. The search engine therefore enforces rules as to the combination of indexed and unindexed terms that may be searched. Any unindexed term may be searched in combination with a result set, with an indexed term or with an intermediate result. Otherwise, if query optimization would result in an unindexed term on its own (or a combination of unindexed terms) forming an SQL statement, a user-authentication parameter obtained during initialization is examined to determine whether the user has the privilege to initiate searches of unindexed terms (very few users will have such permission). If the user does not have permission, the entire search is failed and an appropriate diagnostic is returned.

To allow users to search for multiple values of the same unindexed element (e.g. language English or French or German) in a single query without having to input a query with complex nesting of parentheses and repetition of terms, the search engine allows lists of such values to be sent as a single term, which the engine processes as an SQL "... IN (value,...,value)" statement, e.g., code-language = eng, fre, ger. Queries of this form are handled very efficiently by the RDBMS.

AMICUS Clients

Three separate user interfaces have been created for the AMICUS system. One is a Microsoft Windows interface intended for use by cataloguing and other technical services staff; one, *Access AMICUS*, is a VMS host-based interface using a command-driven paradigm, intended for use by subscribers to the NLC’s MARC record distribution, union catalogue and ILL services; and the last, *ISAAC*, developed for CISTI, is also a VMS host-based interface but using a menu and form-driven paradigm, and intended for use by patrons of the NLC’s and CISTI’s reading rooms and reference services. There are French and English language versions of each of these interfaces, and users can switch freely between the two languages.

The Windows interface provides access to all AMICUS functionality, including bibliographic searching and cataloguing. Access AMICUS supports bibliographic searching, ILL requesting and notification of holdings. ISAAC supports bibliographic searching and the creation of requests for the NLC’s ILL system and CISTI’s document delivery system.

All the clients support complex searching using nested Boolean operators, keyword searching and proximity searching. They all support the use of at least the previous result set as an operator in the query. They all support index scanning and generation of searches from index terms and offer multiple record display formats that are user selectable.

Each of these interfaces uses the same Z39.50 origin software, based on the IR Toolkit developed by Software Kinetics Ltd. The use of the Z39.50 protocol is, however, invisible to users of the AMICUS interfaces.

The interfaces initialize a Z39.50 session when they are started. Subsequently they use the Search service only. The Z39.50 client portion of each interface receives a Z39.58 Common Command Language (CCL) query string from the user interface, together with other search parameters (such as the name of the database to search), parses the query into a Z39.50 type 1 or 101 query, and generates a SearchRequest message which is wrapped inside an AMICUS Protocol message and sent to the search engine. The CCL query string may be either input directly by the user or generated by the user interface in response to mouse clicks or form filling.

None of the clients currently request records to be returned from the Z39.50 search. Instead they access the result set in the Ingres database directly to obtain display information, using the proprietary IngresNet protocol. If a search results in a single record, it is displayed to the user in the session default display format. Numerous predetermined display formats are currently supported by AMICUS, including brief and full labelled formatted displays for specific uses such as interlibrary loan or reference services, and two CanMARC displays, one oriented toward descriptive cataloguers and one for subject cataloguers. If a search results in multiple records, a tabular display of result set records is presented, from which the user can select one or more records for a fuller display.

Similarly, index scanning and result set sorting are implemented through IngresNet access to the Ingres database. When the architectural model for the client was prepared, the 1995 Z39.50 Scan and Sort services were insufficiently stable to permit their use in the AMICUS clients. Future versions of the client may implement the Scan and Sort services.

Each of the clients contains a query parser that generates Z39.50 queries from user input. The same parser is implemented on all clients. This parser accepts a Z39.58 Common Command Language (CCL) string as input and generates as output a Z39.50 version 2 query of type 1 (or 101 if the query contains a proximity operator). The parser uses a table to control the mapping of CCL index names to Z39.50 attribute combinations. This table specifies an attribute value combination for every index name recognized by the parser. Thus, for the Title Keyword index, ("TW") the table specifies the following attribute values: Use = 4, Position = 3, Completeness = 1. Values of other attributes are shown as 0, meaning that the parser should calculate correct values to send on the basis of the query term. If the term contains multiple words, the parser will set the Structure attribute to 1, for "phrase"; if the term contains a final "?", the parser will set the Truncation attribute to 1 for "right truncated", etc. A calculation is not specified for every case; if there is no calculation, the query omits the attribute type completely, allowing the server to use its default value for that attribute. If the input term contains no explicit truncation mark, for instance, no truncation attribute will be contained in the Z39.50 query for that term and the default value of the server will be used. For the AMICUS server the default value for truncation is 100, "do not truncate".

The index table is maintained as an Ingres table, from which configuration files are extracted for use by the clients. For speed at startup, the index table is maintained as two files that are read when the user interface is initialized. The clients implement a paradigm of *indexes* and *limiters*, with "indexes" referring to those searchable elements for which physical indexes are maintained, and "limiters" referring to those elements that are not indexed in the database and are intended to be used only to modify sets created by a search of a primary index (although a few users have permission to search directly on limiters, as described above). There is an index name corresponding to almost every distinct element in the AMICUS data model.

Query strings of almost unlimited complexity can be built through the use of parentheses in the CCL query string, with the limitation that a single query may not exceed 1000 characters and a single term may not exceed 200 characters. Multiple index names can be applied to a single term (e.g. "tw sw nuclear physics"), which the parser interprets as a Boolean OR of the single term applied to the each of the index names ("tw nuclear physics OR sw nuclear physics").

Proximity searching is fully supported by the clients, using the CCL "within" operator for ordered proximity and the CCL "near" operator for unordered proximity. Searchers may use the client's default value for distance, may change the default for a session or may supply a specific value on a case-by-case basis. As mentioned earlier, only the "character" unit is supported for distance.

The Windows client is designed to support searching of both the AMICUS databases and databases at remote servers via a Z39.50 client gateway on the AMICUS server. To support this, the user can choose which system to connect to, and which database at the system to search. The index tables support this model by maintaining different attribute combinations for each index name depending on the system and the database being searched. Only searching of the AMICUS databases will be supported for the initial release of the AMICUS clients, however.

The Windows Interface

The Windows interface permits the searcher to construct queries either by typing in a CCL query string directly or by choosing the various elements of the query from selection lists. It is expected that frequent users of the interface will normally prefer to enter their queries directly, while occasional users will prefer the point-and-click approach. When the searcher uses a selection list to choose an index that requires system-controlled term values, such as a language code or physical description code, a list of the allowed values is presented to the searcher. Selecting a value inserts the index name and value in the query. The searcher can type in operator names directly or click on buttons to select them.

Every AMICUS user has a default database name to which all searches are applied. The searcher can select from the list of AMICUS databases a different database to search. This selection applies for the remainder of the session, or until changed again.

To simplify the input of complex repeated queries, a user can save all or portions of queries for subsequent insertion into a query string. The searcher selects from a list of these "Search qualifiers" and the user interface includes functions that allow a user to manage his or her personal list.

The interface maintains a log of searches that have been performed in the current session. The searcher can examine the log; display records from any previous search; select a previous result set to use as an operand in a subsequent search. The 25 most recent queries are maintained by the log, and the searcher can delete individual queries from the log as desired.

The Access AMICUS Interface

The Access AMICUS interface presents a host-based command-oriented search interface to the National Library's Search Service subscribers across Canada. This interface is deliberately designed to resemble the DOBIS search interface, to minimize the amount of retraining required by the 500-plus Search Service users. The query language is uses the CCL Find and Scan commands, with the exception that single-character commands, "F" for "find" and "S" for "scan", are required, as opposed to the full name and three-character acronyms specified by CCL.

Users of Access AMICUS enter CCL commands to scan indexes and search the AMICUS databases. The interface also allows users to download CanMARC records for cataloguing purposes, notify the National Library of local holdings and make ILL requests for items in the National Library's and CISTI's collections.

The interface does not support a number of the features of the Windows interface: search logs are not maintained; search qualifiers are not available; pop-up lists of term values are limited to only the most commonly used indexes and have limited sets of values. The interface is not currently designed to allow searching of systems other than AMICUS.

The ISAAC Interface

The ISAAC interface is designed for use by untrained end-users of the CISTI and National Library collections. The interface is implemented as a set of menus that lead to search templates, with each template presenting the user an input form that, when completed, specifies a search to be performed. There are templates for performing a "quick search" (name, title and/or distinguishing number such as ISBN or AMICUS number); for searching for monographs, serials, and a number of special types of material such as technical reports, theses, newspapers, audio-visual material, music; and for doing subject searching. Each of these templates uses a dialogue tailored for the specific type of material identified by the template. The interface uses the template and data supplied by the searcher to construct a CCL query string which is passed to the client query parser for generation of a Z39.50 SearchRequest.

When records have been found the user can initiate a dialogue to narrow the search, broaden the search or to "find more like this".

ISAAC offers access to only the National Library collections, CISTI collections and Union Catalogue databases. Only a limited subset of AMICUS indexes is used.

Since the National Library and CISTI are both closed-stack libraries, ISAAC allows the user to issue a request for delivery of an item to a reading room or (for CISTI staff) to an office; it allows the user to issue ILL and document delivery requests to CISTI or the National Library.

Conclusion

This paper has described how Z39.50 provides the core functionality of the AMICUS bibliographic search engine. The server has been designed to provide maximum efficiency in searching and offers subsecond responses in building result sets for common queries. It is designed to be highly flexible and can be adapted to different data models with relative ease. The underlying database technology can be changed without requiring major alterations to the engine, and the system can be ported to different hardware/software platforms with little difficulty.

Z39.50 is used to provide transparent search access to heterogeneous database systems and also to provide access to the National Library of Canada's core information to the widest possible range of clients. AMICUS demonstrates that Z39.50 can successfully be used to provide a public search interface to SQL databases.

One positive effect of the client-server technology used has been the feasibility of creating multiple user interfaces for different purposes, as shown by the three interfaces created for AMICUS.

AMICUS is a significant new bibliographic system, designed from the ground up to meet the needs of two large research institutions, one of which has the additional requirements of a national library.